FLEXPOP: A POPULARITY-BASED CACHING STRATEGY FOR MULTIMEDIA APPLICATIONS IN INFORMATION-CENTRIC NETWORKING

IKRAM UD DIN

DOCTOR OF PHILOSOPHY
UNIVERSITI UTARA MALAYSIA
2016

Perakuan Kerja Tesis/Disertasi

(To be substituted with signed document for this page)

Permission to Use

Declaration Associated with This Thesis

Some of the work presented in this thesis have been published or submitted as listed below:

1. **Ikram Ud Din**, Suhaidi Hassan, Adib Habbal, and Nur Haryani Zakaria, "A Popularity-based Caching Strategy for the Future Internet," ITU Kaleidoscope 2016, (To Appear).

2. **Ikram Ud Din**, Suhaidi Hassan, and Adib Habbal, "A Content Placement Scheme for Information-Centric Networking," *Advanced Science Letters*, 21(11), pp. 3484–3486, 2015.

3. **Ikram Ud Din**, Suhaidi Hassan, and Adib Habbal, "Redundancy Elimination in the Future Internet," *Lecture Notes in Computer Science, Springer*, 2016 (To Appear).

4. **Ikram Ud Din**, Suhaidi Hassan, and Adib Habbal, "SocialCCNSim: A Simulator for Caching Strategies in Information-Centric Networking," *Advanced Science Letters*, 21(11), pp. 3507–3511, 2015.

5. **Ikram Ud Din**, Suhaidi Hassan, and Adib Habbal, "Comparison of Caching Strategies on Different Topologies in Information-Centric Networking," *Fourth International Conference on Internet Applications, Protocols and Services* (NETAPPS2015), pp. 37-41, 2015.

6. **Ikram Ud Din**, Suhaidi Hassan, and Adib Habbal, "A Content Eviction Mechanism for Information-Centric Networking," *ARPN Journal of Engineering and Applied Sciences*, 11(5), pp. 3233-3235, 2016.

7. **Ikram Ud Din**, Suhaidi Hassan, and Adib Habbal, "A Mechanism for Reducing

Content Retrieval Delay in the Future Internet," *Fourth International Conference on Internet Applications, Protocols and Services* (NETAPPS2015), pp. 59-62, 2015.

Abstrak

Perangkaian Bertumpuan Maklumat (ICN) adalah senibina Internet masa hadapan yang dominan. Dalam ICN, item kandungan disimpan seketika oleh nod-nod rangkaian seperti penghala. Apabila memori penghala penuh dan tiada lagi ruang untuk kandungan yang baharu tiba, kandungan yang disimpan akan dikeluarkan untuk menangani saiz ruang simpanan penghala yang terhad. Oleh yang demikian, adalah penting untuk membangunkan satu strategi penyimpanan yang berkesan bagi menyimpan kandungan popular untuk jangka masa yang lebih lama. Kajian ini mencadangkan satu strategi penyimpanan yang baharu, dinamakan Penyimpanan Lentur Berdasarkan Populariti (*FlexPop*) bagi penyimpanan kandungan popular. FlexPop mengandungi dua mekanisme iaitu Mekanisme Penempatan Kandungan (CPM) yang bertanggungjawab untuk penyimpanan kandungan dan Mekanisme Pengusiran Kandungan (CEM) yang menangani pengusiran kandungan apabila simpanan penghala penuh dan tiada lagi ruang untuk kandungan yang baharu tiba. Kedua-dua mekanisme ini disahkan melalui Teori Set Kabur, mengikut Kaedah Penyelidikan Reka Bentuk (DRM) yang digunakan dalam penyelidikan ini bagi memastikan kerapian kerja serta keboleh-ulangan di bawah keadaan setanding. Prestasi FlexPop dinilai menggunakan simulasi dan kemudiannya dibandingkan dengan keputusan bagi strategi-strategi Tinggalan Salinan Merata (LCE), ProbCache serta strategi Kandungan Paling Popular (MPC). Keputusan kajian menunjukkan bahawa strategi FlexPop melebihi jangkuan strategi LCE, *ProbCache* dan MPC dari segi kadar sasar simpan, kelewahan, kelengahan perolehan semula kandungan, penggunaan memori dan regangan, yang merupakan metrik-metrik penting (dalam kebanyakan kajian) untuk tujuan penilaian penyimpanan ICN. Hasil yang dipamerkan dalam kajian ini adalah penting untuk membolehkan *FlexPop* diterima oleh pengguna kerana dengannya mereka mampu menentusahkan prestasi ICN sebelum memilih strategi penyimpanan yang sesuai. Justeru, *FlexPop* berpotensi dalam penggunaan ICN untuk Internet masa hadapan umpamanya bagi pengerahan teknologi IoT.

Kata kunci: Perangkaian bertumpuan maklumat, Internet masa hadapan, *FlexPop*, Penempatan kandungan, Pengusiran kandungan

Abstract

Information-Centric Networking (ICN) is the dominant architecture for the future Internet. In ICN, the content items are stored temporarily in network nodes such as routers. When the memory of routers becomes full and there is no room for a new arriving content, the stored contents are evicted to cope with the limited cache size of the routers. Therefore, it is crucial to develop an effective caching strategy for keeping popular contents for a longer period of time. This study proposes a new caching strategy, named Flexible Popularity-based Caching (FlexPop) for storing popular contents. The FlexPop comprises two mechanisms, i.e., Content Placement Mechanism (CPM), which is responsible for content caching, and Content Eviction Mechanism (CEM) that deals with content eviction when the router cache is full and there is no space for the new incoming content. Both mechanisms are validated using Fuzzy Set Theory, following the Design Research Methodology (DRM) to manifest that the research is rigorous and repeatable under comparable conditions. The performance of FlexPop is evaluated through simulations and the results are compared with those of the Leave Copy Everywhere (LCE), ProbCache, and Most Popular Content (MPC) strategies. The results show that the FlexPop strategy outperforms LCE, ProbCache, and MPC with respect to cache hit rate, redundancy, content retrieval delay, memory utilization, and stretch ratio, which are regarded as extremely important metrics (in various studies) for the evaluation of ICN caching. The outcomes exhibited in this study are noteworthy in terms of making FlexPop acceptable to users as they can verify the performance of ICN before selecting the right caching strategy. Thus FlexPop has potential in the use of ICN for the future Internet such as in deployment of the IoT technology.

Keywords: Information-centric networking, Future Internet, FlexPop, Content placement, Content eviction

Acknowledgements

In the name of ALLAH, the Most Gracious, the Most Merciful:

"*As for those who strive hard in Us (Our Cause), We will surely guide them to Our Paths. And verily Allah is with the good doers.*"

(Al-Quran - 29:69)

I would like to acknowledge those individuals without whose help this thesis could not have been completed by me. First, I thank my supervisors: Prof. Dr. Suhaidi Hassan and Dr. Adib Habbal. Their assistance throughout the research and supervision during the writing were invaluable. Apart from my devotion, their professionalism, daily meetings, and progress inspection were the main reasons of success. Working with them in the field of ICN increased my understanding and improved my intellection. May Allah give them its reward and make them happy and prosperous in the rest of their life. Next, I would like to extend my thanks to the Chairperson of my viva session: Prof. Dr. Zulaikha Jamaluddin, my thesis reviewers: Prof. Dr. Hj. Mazani Manaf and Associate Prof. Dr. Osman Ghazali for their insightful comments and guidance. I would also like to thank my team members: Abdullahi Ibrahim as forever friend and ICN team member, Mohamed Firdhous for helping me in LyX and Latex during thesis writing, and Dr. Cesar Bernardini for guiding throughout the strenuous exercises of SocialCCNSim simulations that will never be forgotten, and the whole group of InterNetWorks Research Laboratory for their motivation all along this research. Finally, I would like to thank my parents, brothers, sisters, and my wife for their support and devotion through all the process of researching and writing this thesis.

Dedication

Dedicated to my whole family . . .

List of Abbreviations

AS	-	Autonomous System
CCN	-	Content-Centric Network
CDN	-	Content Delivery Network
CEM	-	Content Eviction Mechanism
CLS	-	Chunk Caching Location and Searching Scheme
COMET	-	Content Mediator Architecture for Content Aware Network
CPM	-	Content Placement Mechanism
CR	-	Content Router
CS	-	Content Store
CT	-	Comparison Table
DHT	-	Distributed Hash Table
DONA	-	Data Oriented Network Architecture
DoS	-	Denial of Service
DRM	-	Design Research Methodology
DS	-	Descriptive Study
FGPC	-	Fine-Grained Popularity-based Caching
FIB	-	Forwarding Information Base
FIFO	-	First-In-First-Out
FlexPop	-	Flexible Popularity-based Caching
ICN	-	Information-Centric Network
IP	-	Internet Protocol
LAN	-	Local Area Network
LCD	-	Leave Copy Down
LCE	-	Leave Copy Everywhere
LFU	-	Least Frequently Used
LNC	-	Linear Network Coding
LRU	-	Least Recently Used
MADM	-	Multiple Attribute Decision Making
MAUT	-	Multiple Attribute Utility Theory

MCD	-	Move Copy Down
MPC	-	Most Popular Content
NAT	-	Network Address Translation
NCCM	-	Network Coding based Cache Management
NDN	-	Named Data Network
NDO	-	Named Data Object
NetInf	-	Network of Information
PIT	-	Pending Interest Table
PS	-	Prescriptive Study
PSIRP	-	Publish-Subscribe Internet Routing Paradigm
PT	-	Popularity Table
PURSUIT	-	Publisher Subscriber Internet Technology
P2P	-	Peer-to-Peer
RC	-	Research Clarification
SAIL	-	Scalable Adaptive Internet Solution
SAW	-	Simple Additive Weighting
TLRU	-	Time Aware Least Recently Used
TOPSIS	-	Technique for Order Preference by Similarity to Ideal Solution
TSB	-	Time Since Birth
TSI	-	Time Since Inception
VoD	-	Video on Demand
WAN	-	Wide Area Network

Table of Contents

List of Tables

List of Figures

CHAPTER ONE
INTRODUCTION

The existing Internet was intended to address the correspondence demands of a period when a communication network was required to share expensive and rare resources, such as long distance communication links, peripherals, and mainframe computers [1]. The main design rules of the Internet made it possible to connect new systems to the Internet and allowed a remarkable development in its size. However, the inspiring growth of the Internet has offered ascend to new requirements from the design, for example, content dissemination, storage resources, quality of service, mobility, security, scalability, and economics [2]. Moreover, due to its particular implementation and end-to-end approach, the current Internet architecture has many limitations. For example, a variety of add-on patches, such as Point-to-Point (P2P) overlays, Content Delivery Networks (CDNs), Mobile IP, and Network Address Translation (NAT), that were not part of the original design, all violate, in different manners, various features of the initial Internet architecture [3].

Furthermore, the initial protocols and Internet architecture were designed expecting a cooperative and acceptable environment, which is far away from practicality, where lack of privacy and security threats, for example, phishing, Denial of Service (DoS) attacks, and malware, have turned out to be progressively predominant [4]. It has been recently perceived that information-centric usage of the Internet has gained popularity as revealed by the majority being associated with it [5]. This usage brings up a range of design challenges, several of which were not successfully handled by the existing architecture [6]. These challenges include information scarcity, security and liability through controlled information distribution, tussle mediation through information governance, and medium-independent information access.

Here, a question has been raised that whether the Internet needs a new clean-slate design method or we can keep "patching over patches," [7]. Therefore, a research group [8] has been formed to identify the impediments of the existing Internet and discuss the major objectives and requirements of the future Internet architecture. In this situation, Information-Centric Networking (ICN) has turned into a favorable nominee for the future Internet architecture.

1.1 Motivation

Network operators have been waiting for data tsunami since mid-1990s, which luckily did not come, but due to exponential growth in the number of Internet users and interest for multimedia applications, the alarm of data tsunami is returning [9, 10]. The Cisco Visual Networking Index [11] exhibits that the delivery of multimedia applications has become extremely popular on the Internet. Surprisingly by 2017, global IP traffic will be 110.3 exabytes per month or 1.3 zettabytes per year. The widespread Internet multimedia traffic will reach 55 percent of all user Internet traffic, which was 51 percent in 2011. In addition, digital devices created four zettabytes of data in 2013, while it is predicted that the number of communicating devices will be three times more than the world population by 2017. The aggregation of all kinds of videos, i.e., video on demand (VoD), Point to Point (P2P), and Television (TV) will be roughly 86 percent of the worldwide user traffic. Currently, 20.8 million messages are shared every single minute only on WhatsApp which will cross 270 million by 2017 [12]. This evolution is more vividly depicted in Figure 1.1 which represents the number of data and its type that is transferred in an Internet minute [13].

Figure 1.1 demonstrates a strong hold of how the present host-to-host Internet is overloaded with levels of popularity of information. As indicated by the most recent data traffic and information sharing on the Internet in a minute [11, 13, 14], an approx-

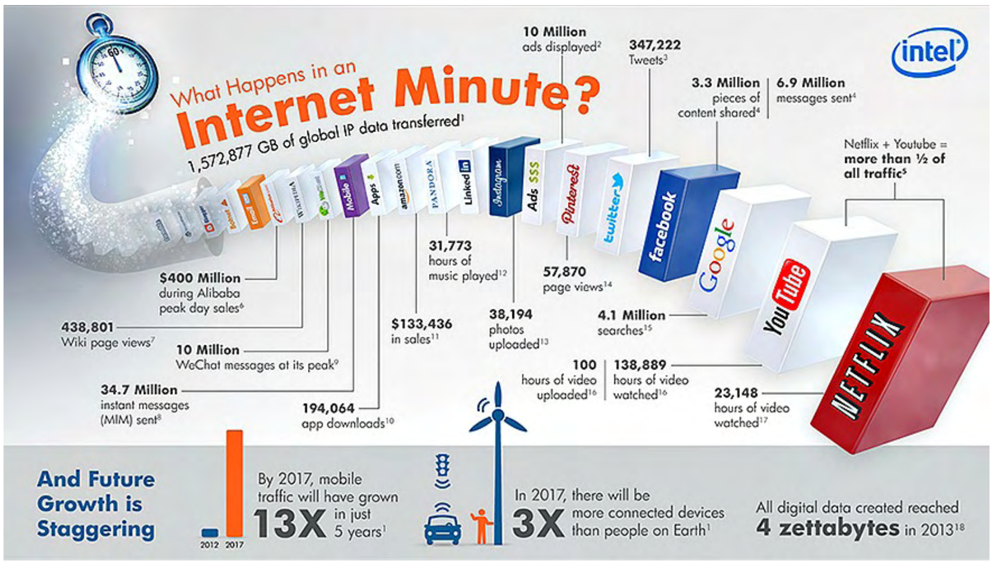


Figure 1.1. Internet in a Minute [13]

imated more than 34.7 million messages are sent over the Internet. With user diversification tending towards multimedia applications, popularity of videos shared on YouTube has recorded a stunning 138,889 hours of video viewed against an estimated 100 hours of videos uploaded every single minute. This makes up more than half of the overall Internet traffic when YouTube and NETFLIX videos are collectively calculated. Furthermore, mobile traffic will reach 13 times more than the current traffic by 2017. With this data calculation, where it is noticeable that maximum shared/accessed data is multimedia, it is worthy to dissociate host-centrism of data to an information-centric paradigm. This may be easily manageable by flexibly placing caching nodes in the network through ICN concepts.

According to the ICN Research Group (ICNRG) at the Internet Research Task Force (IRTF) [8], data manipulation and distribution have been the business of the current Internet. Therefore, it is indispensable to diminish the challenges causing threats to the future Internet. Among the objectives of the ICNRG, the recent drafts [15, 16] include: the proposal and creation of scalable routing schemes, name resolution, cache management, and performance evaluation of the ICN paradigm. These along other challenges make ICN caching noticeable to conform with the recent drafts and recommendations of ICNRG. Moreover, Google being utilized as one of the leading search engines, received a maximum number of hits, i.e., 4.1 million searches in an Internet minute [13]. Part of the motivation is that, when queries are served, a local caching node may have a copy in its content store so that if a new request arrives for that content, it can be satisfied locally rather than forwarding to the main server. This approach will increase content hit rate and reduce bandwidth utilization as well as content retrieval delay. In this case, ICN has gained such a popularity to be considered a vowing nominee for the future Internet architecture.

1.1.1 Background of ICN Design

The configuration of ICN is one of the notable aftereffects of various global future Internet research activities. The current Internet architecture depends on end-to-end correspondence between hosts, which is called host-centric networking. Instead, the growing demand for efficient and highly scalable distribution of contents (for example, documents, videos, web pages, or other types of information) has motivated the modification of architectures that concentrate on Named Data Objects (NDOs). The architectures of ICN can influence multiparty correspondence through replication, in-network caching, and interaction models such as publish-subscribe to achieve effective and consistent transmission of contents by providing a common platform for communication services [17]. The ICN approach is being investigated by various research projects, including the US funded projects: Data Oriented Network Architecture (DONA) [18], Named Data Networking (NDN) [19], MobilityFirst [20], Content-Centric Networking (CCN) [21], and the European funded projects: Publish-Subscribe Internet Routing Paradigm (PSIRP) [22], 4WARD [23][24], Publish-Subscribe Internet Technology (PURSUIT) [25], CONVERGENCE [26], Scalable & Adaptive Internet soLutions (SAIL) [27], and COntent Mediator architecture for content-aware

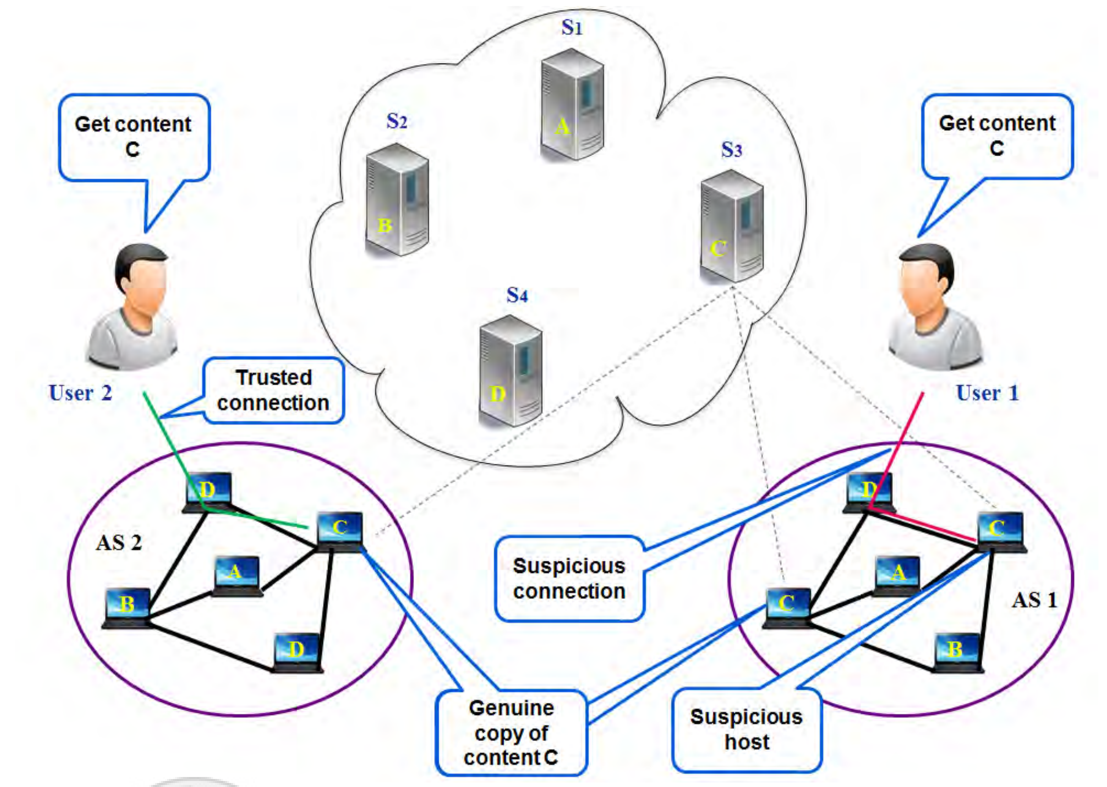


Figure 1.2. ICN Communication Model: Adapted from [29]

nETworks (COMET) [28]. While the approaches of these projects differ concerning their specific design, they share some architectural properties, objectives, and suppositions [17].

Generally, the purpose is to create system models that are appropriate for content distribution and that better adapt to flash-crowd impacts, disruptions, and disconnections in the communication service [29]. Communication is driven by recipients *requesting* data objects. Senders publish the objects to make NDOs available to receivers.

In Figure 1.2, the network is divided into autonomous systems (AS), i.e., AS1 and AS2, respectively. A node in AS2 has a genuine copy of content *C*, accessed by a trusted User2 through a trusted connection. Another node in AS1 has a copy of the same content, accessed through a suspicious connection. The figure has four servers: S1, S2, S3, and S4. The network can assure the request with information from any node having a copy of the item, enabling application-independent and effective caching as a component of the system administration. This decoupling in time and space between publisher and subscriber needs that data objects carry security metadata for confirming the authenticity and integrity of the objects. The ICN research community accepts that named data, rather than its physical location, is the key component of routing in ICN. One of the leading proposals, CCN [30], has entirely changed information transmission by sending content requests and receiving through name at the network layer itself.

1.1.2 The Main Components of ICN

The ICN architecture, as shown in Figure 1.3 [31], is different from that of the IP architecture. The existing Internet architecture follows a source-driven approach, i.e., a connection is established between the source and destination before any data transfer. The main objective of ICN is to find, transfer, and distribute contents as opposed to the reachability of end users and the establishment of conversation among them. In ICN, the customer requests content without knowing the providing host, and the route is established by the receiver to the customer, i.e., the communication follows a receiver-driven approach and the data follows the reverse route. The system is then responsible for mapping the requested data and its location. One of the applicable aspects of ICN for providing efficiency is naming, i.e., content must be named such that it should be independent of the node providing the content.

The most popular and leading property of ICN is caching, where the router can store the content (for a time) passing through it (this storage depends on the cache size and replacement strategy) and provide them to the requesting clients. In ICN caching, the content is replicated through a caching strategy, therefore, the possibility of content delivery to the end user(s) is increased. Generally, the content in ICN is requested by

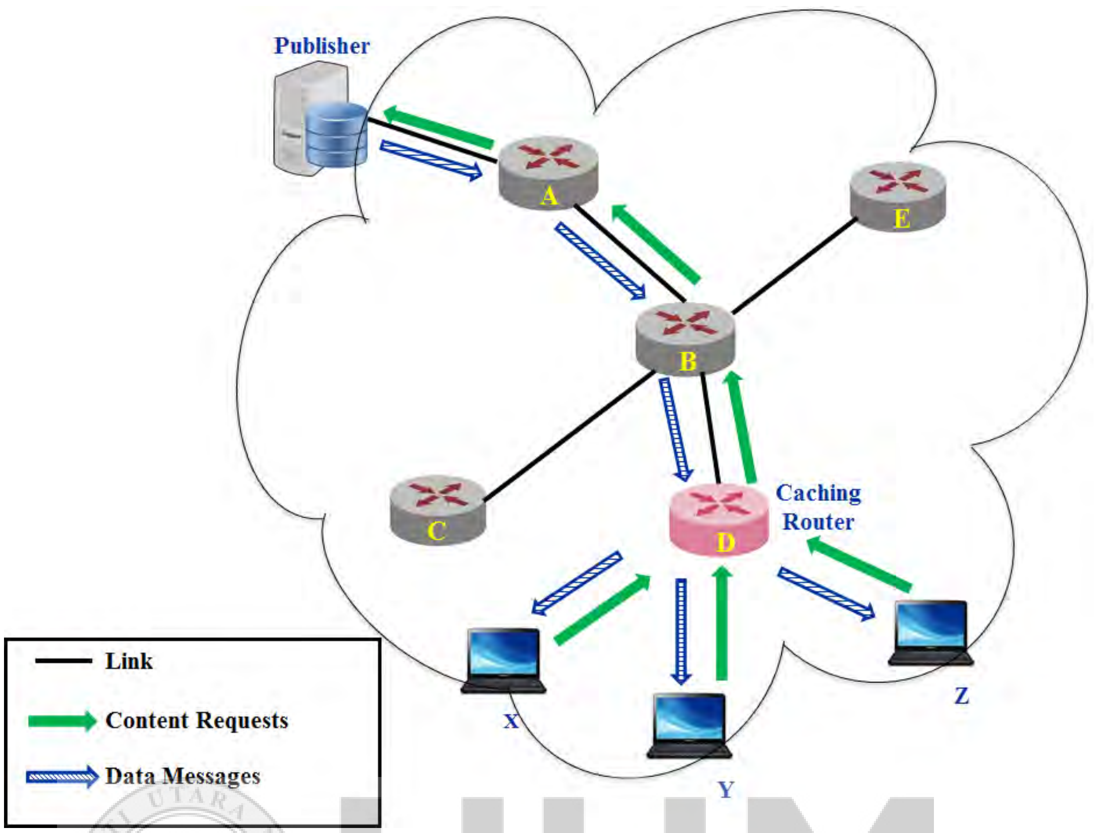


Figure 1.3. An Information-Centric Network: Adapted from [31]

user(s) without having information of the providing host. This is based on the receiver-driven approach, i.e., the path is established by the publisher to the subscriber and the data is forwarded on the reverse path. Mapping between the requested content and its location is then the responsibility of the network [31].

1.2 Problem Statement

The ICN network is intrinsically classified as a multi-stage caching system, where the customers generate requests from geographically dispersed nodes and the caches are physically located across the network. When a content item is requested from an intermediate node, then if a router has a cached copy, the request is satisfied locally. This leads to the design problems of effective caching strategies such as content placement and best possible position of caches. In other words, a popularity-based caching strategy is needed to improve the utilization of limited cache space and reduce the latency of user content requests. The latency of data request is smaller when more popular contents are cached at the node which is closer to the requesting user(s). However, due to the limited cache space, the nearest node is unable to cache all contents, thus an efficient allocation/placement mechanism is required to cache contents at a suitable location. Likewise, if the cache of a content store becomes full and a new content arrives then the new content will replace the old one. Here a question arises that which content will be evicted from the cache to accommodate the new arrived one. This process takes some time to replace content from the cache, and hence leads to the searching overhead during replacement process. Therefore, a content eviction mechanism is needed to evict the content and place it at some other location so that the stretch ratio should be reduced and the content can be accessed (without extra delay) if a new request arrives for that.

For that, a strategy named Leave Copy Everywhere (LCE) [30, 32] was proposed which is also called the default-ICN strategy, where every router caches a copy of content locally. In this strategy, due to redundant copies of the same content, at some time, the caches of all network nodes become full and no space is left for the new incoming content. As a result, the *cache hit rate*, which is one of the prominent metrics in the evaluation of ICN performance, is drastically reduced.

To overcome the problems in LCE, i.e., to reduce *redundancy* and increase *cache hit rate*, several researchers designed numerous caching strategies, for example, popularity-based caching [33, 34, 35, 36, 37], probabilistic caching [38, 39, 40, 41], caching for bandwidth optimization [42, 43], caching for redundancy elimination [44], caching for content distribution [45, 46, 47, 48], and some other useful ones [49, 50, 51, 52, 53, 54, 55, 56, 57]. However, the most dominant of them is the Most Popular Content (MPC) [33] because it has performed better than many other strategies [58].

Even though MPC outperforms some of the existing strategies [58], it has some undesirable features, for example, it utilizes maximum memory when it shares its popularity table with all neighbor routers. Actually, the contents are cached in the random access memory (RAM), however, the available static random access memory (SRAM) or dynamic random access memory (DRAM) are limited in the size. The DRAM, which is a volatile memory and needs to be refreshed regularly [59] is currently available in 10GB maximum [60, 61]. And the popularity table in MPC consists of all incoming requests regardless of their popularity value, therefore can occupy maximum space in the memory. Moreover, if the memory of router becomes full and a new content arrives, it does not have any policy for that but simply replaces one of the cached contents by the new arrived one using Least Recently Used (LRU) policy. As LRU replaces the content based on the access time, the replaced content may be very popular and therefore the subsequent requests for the same content will be forwarded to the server. In this way, extra content retrieval delay may occur and thus maximum bandwidth is utilized.

1.3 Research Questions

In addressing ICN cache management, the following research questions were raised up:

1. How to cache the requested content items in content store in order to increase cache hit rate and reduce caching redundancy?
2. How to minimize memory consumption and stretch ratio during content placement and eviction?
3. What is the impact of the proposed strategy on cache hit rate, redundancy, re-

trieval delay, memory consumption, and stretch ratio?

1.4 Research Objectives

The purpose of this research is to design a new cache management strategy to achieve better performance in terms of cache hit rate, content retrieval delay, memory utilization, and stretch ratio during content placement and eviction. This ambition can be further elucidated by the following specific research objectives:

1. To design a popularity content placement mechanism using on-path caching to improve caching gain and reduce both content delay and redundancy of the network caching.
2. To develop a content eviction mechanism by exploiting request records for effective memory utilization and stretch ratio.
3. To evaluate performance of the proposed system using cache management schemes and compare with the existing strategies in simulated network environment.

1.5 Research Scope

The intention of this research is to propose a flexible caching strategy for ICN. More specifically, the focus is on the design of mechanisms appropriate for increasing cache hit rate and reducing path redundancy during content caching and eviction. In addition, the proposed strategy, named Flexible Popularity-based caching (FlexPop), efficiently utilizes memory in terms of convergence and retrieval delay of contents. Moreover, the FlexPop strategy is designed to take into consideration the most demanding data, i.e., multimedia applications. Examples of multimedia applications, which are requested and/or used by million of users everyday, are video-on-demand (VoD), distance learn-

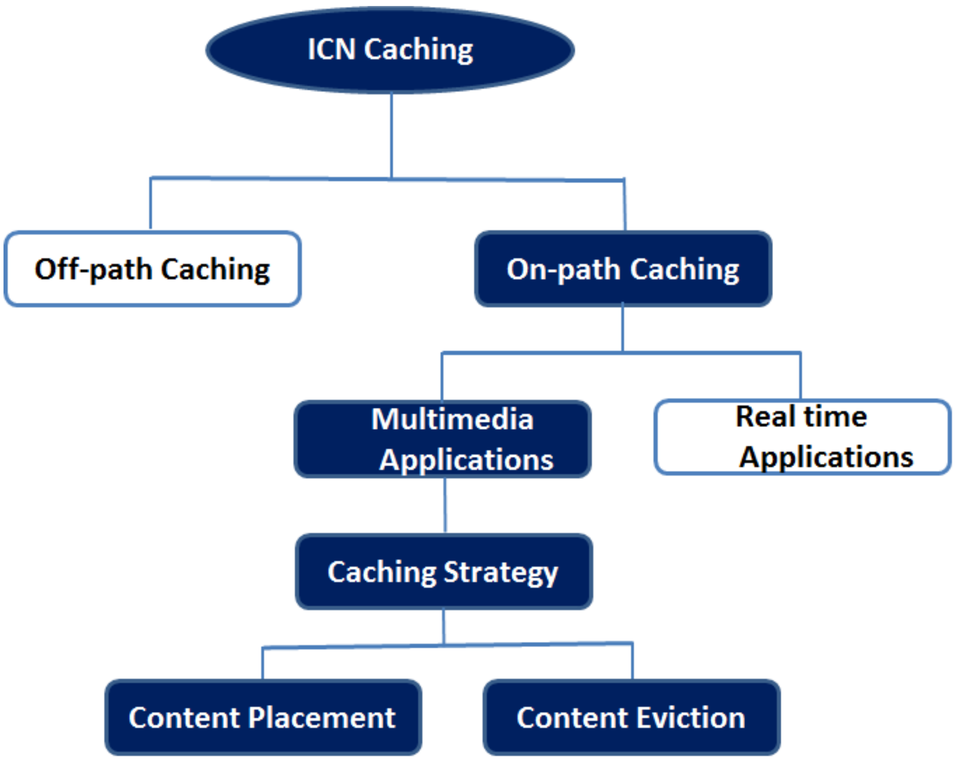


Figure 1.4. Scope of the Study

ing, patient monitoring systems, remote robotic agents, interactive games, and so on. The scope of this research is presented in Figure 1.4. In the given figure, the ICN caching is divided into two parts, i.e., off-path caching and on-path caching. The proposed caching strategy is designed for multimedia applications and comes under the on-path caching.

1.6 Research Steps

In order to accomplish the main objectives of this research, the following fundamental steps were performed:

1. Survey of the existing content placement and cache management strategies to find out the ICN challenges and future research directions.
2. Propose the deterministic and consistent case scenarios of content generation utilizing satisfactory statistical distribution.

3. Explore the properties of *Simple Additive Weighting* (SAW) and its flexibility to content placement.
4. Investigate the characteristics of *Technique for Order Preference by Similarity to Ideal Solution* (TOPSIS) and its coherence to content eviction.
5. Comparative performance evaluation of the current ICN caching approaches and the proposed FlexPop strategy to investigate their contributions and limitations.

1.7 Research Significance

This study aims to be significant for advancing and formalizing the ICN infrastructure. Moreover, the study will likewise be essentially useful in realizing the Internet of Things (IoT) and Internet of Everything (IoE) due to the design that provides the Internet ubiquity. The outcomes of this research will be used to assist content distribution on the Internet considering the significance in reliable data transfer that may lead to a set of other applications required by users. If it is effectively investigated in other network domains, for example, Grid computing and Cloud, the FlexPop strategy will improve due date realization of the former in data transmission and enhance assurance in the latter when the Infrastructure as a service (IaaS) and software as a service (SaaS) among other services are utilized.

Caching affects the nature of high information reach-ability to end-users of the Internet. The current impact of users demanding multimedia applications would mitigate the problems of data flooding, packet loss, bandwidth utilization, and placement of redundant contents in locations that are not needed. When the idea of ICN caching is standardized, the entire ICN research community will benefit from its implementation through:

1. Coming up with novel caching strategies in ICN that will enhance the manage-

ability through careful content placement in the cache of a content store.

2. Proposing new caching algorithms that can be used in ICN approach designs with marginal configuration complexity.
3. Outlining and extending the suitable simulation tools by including the proposed ideas through programming.

1.8 Research Contributions

In the context of ICN, for reliable data transfer and content dissemination, in-network caching plays an important role. For this reason, numerous cache management strategies have been proposed so far. But as ICN is the future of Internet and will probably be implemented by 2020 [62], it is too early to decide that which caching strategy fulfills the ICN requirements. Keeping in mind the basic ICN requirements, this thesis proposes a cache management strategy for storing popular contents on the path from publisher to subscriber. Following are the major significant points that are taken into consideration in this research:

1. Efficient caching algorithm and mechanism that determine the network location of caches. In other words, what content is worthy of caching and what is the suitable location of caches so that the network throughput is enhanced with regard to cache hit rate, path redundancy, and content retrieval delay.
2. The design of a *Content Placement Mechanism* (CPM) that caches popular contents for improving cache hit rate and reducing path redundancy as well as content retrieval delay. As described in Section 1.4, the CPM stores contents at a router which has the highest number of outgoing interfaces, therefore, most of the content requests will pass through that router. Now, if a content is available there, it is satisfied locally rather than forwarding to the publisher. Due to the popularity of a content, maximum requests are expected for that and hence (apart

from improving cache hit rate and reducing content retrieval delay and redundancy) sufficient bandwidth is saved if it is available locally in such a position that maximum requests can go through it.

3. Suggestion of algorithm and mechanism for the supervision of routing processes by persuading the forwarding of requested content to enhance the network performance in terms of memory utilization and stretch.
4. The development of a *Content Eviction Mechanism* (CEM) for flexible content eviction. The term flexible is suggested because unlike other replacement policies it does not delete the content but evicts from the router's cache and stores at another router which has the second number of highest interfaces in the network. This mechanism efficiently utilizes memory and increases *hop decrement* that in turn reduces stretch ratio. The default ICN strategy - LCE, caches every incoming content at each node on the publisher to subscriber path and hence the content stores of routers become full very rapidly.

In contrast, the CEM stores the evicted popular contents at another router having the second maximum number of interfaces, therefore, the content request can be satisfied locally and consequently the turn around time is minimized. Moreover, it increases the *hop decrement* and reduces the stretch ratio because when the selected router has many outgoing interfaces, the distance between that router and the subscriber must be lesser than the distance between the subscriber and the publisher. In case the second highest interfaced router fully utilizes memory then the third highest interfaced router is chosen for caching and thus the contents get closer to the user(s).

Apart from contributions, the proposed strategy also has some limitations. For example, it can perform well in the infrastructure-based (fixed) wireless networks, but as the contents are cached at a node (router) which has the maximum number of outgoing interfaces, it can be difficult to specify such router in other ad hoc networks such as vehicular ad hoc networks (VANETs).

1.9 Thesis Organization

This thesis is organized in six chapters according to the following chapter highlights:

Chapter One starts with the introduction of Information-Centric Networking (ICN) and then discusses the importance of ICN architecture for the future Internet. After that, a comprehensive overview of the thesis is presented with respect to the following points: problem statement of the research, research questions, research objectives, research scope, research steps, and significance of the research.

Chapter Two presents some ICN architectures followed by the importance of in-network caching in ICN. Moreover, it gives a detailed explanation about the two caching approaches, named off-path caching and on-path caching, and explores some existing ICN caching strategies along with their contributions and limitations.

Chapter Three outlines the Design Research Methodology (DRM) as a framework for this study and combines numerous approaches adopted to propose and implement the FlexPop strategy. This chapter explains the following four stages of DRM in full detail: Research Clarification (RC), Descriptive Study-I (DS-I), Prescriptive Study (PS), and Descriptive Study-II (DSII). A diagrammatic representation of the conceptual model for FlexPop is also given in this chapter.

Chapter Four is related to the modeling of FlexPop strategy and its two mechanisms for content placement and eviction, named CPM and CEM, respectively. The chapter discusses the design motivation of FlexPop along with its verification and validation.

The chapter also describes two techniques of fuzzy set theory, based on *Multiple Attribute Decision Making* (MADM), called SAW and TOPSIS.

Chapter Five evaluates the performance of FlexPop and its comparison with the LCE, ProbCache, and MPC through simulations. The simulation metrics for evaluation are cache hit rate, path redundancy, content retrieval delay, memory utilization, and stretch.

Chapter Six expresses the conclusion as well as the contributions and limitations of the research work introduced in this thesis, and then suggests further directions for future studies in line with some pivotal ICN research challenges.

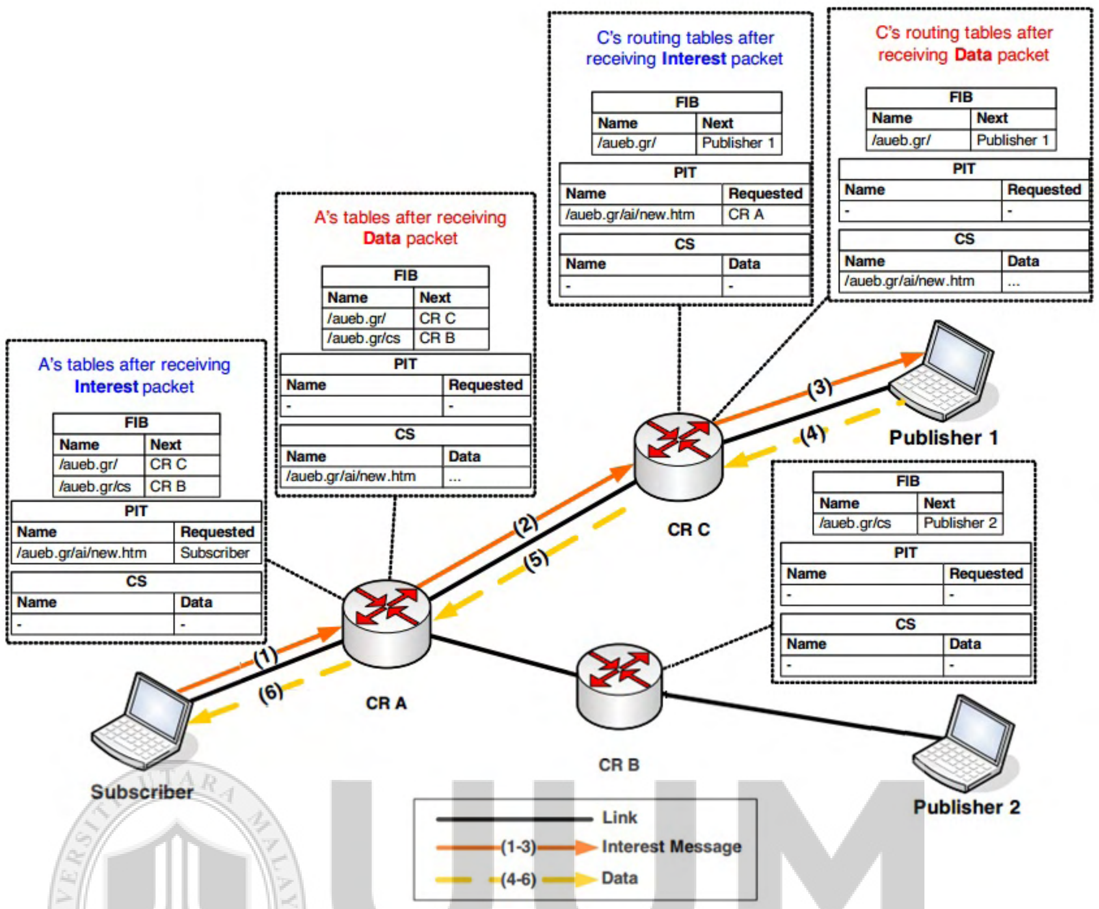
CHAPTER TWO
LITERATURE REVIEW

This chapter explores into greater detail the background and several important past researches related to ICN caching, which would assist in defining the general framework of this research. Some main ICN architectures are presented in Section 2.1 and then in-network caching in ICN is discussed in Section 2.2. Section 2.3 presents some existing cache management strategies along with their contributions and limitations.

2.1 ICN Architectures

Generally, in ICN a customer requests some content and a provider delivers it; the network is responsible for the settlement of request and delivery. ICN architectures consider a variety of design options relating to content name, location, and distribution. This section presents the main idea of ICN architecture, which is based on the surveys presented in [1, 17, 31, 63].

The Named Data Networking (NDN) [21] is a US based funded project that develops the CCN architecture [64]. The workflow of CCN-based ICN approach is shown in Figure 2.1 [1]. Two different types of packets are used for communication in CCN, i.e., INTEREST and DATA packets. To request content items, the subscribers send INTEREST packets, which are received in the form of DATA packets, with both types of packets carrying the name of the requested/responded content item. Content Routers (CRs) then forward all packets hop-by-hop [65] maintaining three data structures: the Forwarding Information Base (FIB), the Content Store (CS), and the Pending Interest Table (PIT). The FIB is used to forward packets to the exact data source, the CS temporarily caches the received data packets, and the PIT tracks a set of interfaces from which pending INTEREST packets have been received. In simple words, it contains


C's routing tables after receiving Interest packet
C's routing tables after receiving Data packet
A's tables after receiving Data packet
A's tables after receiving Interest packet
Publisher 1
Publisher 2
Subscriber
CR A
CR B
CR C
(1)
(2)
(3)
(4)
(5)
(6)
Link
(1-3) Interest Message
(4-6) Data

A's tables after receiving Interest packet

FIB	
Name	Next
/aueb.gr/	CR C
/aueb.gr/cs	CR B

PIT	
Name	Requested
/aueb.gr/ai/new.htm	Subscriber

CS	
Name	Data
-	-

A's tables after receiving Data packet

FIB	
Name	Next
/aueb.gr/	CR C
/aueb.gr/cs	CR B

PIT	
Name	Requested
-	-

CS	
Name	Data
/aueb.gr/ai/new.htm	...

C's routing tables after receiving Interest packet

FIB	
Name	Next
/aueb.gr/	Publisher 1

PIT	
Name	Requested
/aueb.gr/ai/new.htm	CR A

CS	
Name	Data
-	-

C's routing tables after receiving Data packet

FIB	
Name	Next
/aueb.gr/	Publisher 1

PIT	
Name	Requested
-	-

CS	
Name	Data
/aueb.gr/ai/new.htm	...

FIB	
Name	Next
/aueb.gr/cs	Publisher 2

PIT	
Name	Requested
-	-

CS	
Name	Data
-	-

Figure 2.1. CCN-based ICN Architecture [1]

the names of INTEREST packets for which DATA packets are expected.

In the given figure, the subscriber sends an INTEREST for the name /aueb.gr/ai/new.htm (arrows 1-3). If entry with the exact name is found in the PIT, the node discards this INTEREST and records the incoming interface to this PIT, successfully developing a multicast tree for the content item. This packet is sent back to the subscriber(s) in a hop-by-hop fashion, based on the status maintained in the PITs. Especially, upon receiving a DATA packet, a router first caches the matching content item in its CS and then performs a longest-prefix match on its PIT to trace an entry corresponding to the DATA packet. In case the PIT entry records more than one interface, the DATA packet is duplicated, and hence multicast delivery is achieved. Eventually, the router sends the DATA packet to these interfaces and deletes the entry from the

PIT (arrows 4-6). If no matching entry is found in the PIT, the router simply discards the DATA packet as a duplicate.

2.2 In-network Caching in ICN

Caching is one of the crucial components of ICN [66, 67] which borrows its concept from the current Internet presented in [68, 69, 70, 71, 72, 73, 74, 75]. In-network caching is implicitly or explicitly studied in [76, 77, 78, 79, 80, 81, 82]. In this section, the importance of cache management and the two approaches used for in-network caching, i.e., off-path caching and on-path caching, are explored.

2.2.1 Cache Management

To store contents precisely, that may change in time, some improvements are needed in traditional cache systems. One of the possible schemes to name these kinds of contents is to save their names, regardless of the fact that the contents change. If, for example, two routers store the content generated by two different sensors at different times, the information stored at different locations may be inconsistent [83]. Therefore, it is primarily important to deal with the outdated information. In this regard a time-stamp field could contribute, but it is not the proper solution as decades of cache consistency research revealed [84, 85, 86, 87, 88, 89, 90].

Cache management in the time of congestion is a challenging issue in ICN. Reliable data delivery and flow control are usually viewed as considerably more of a challenge in content-centric publish/subscribe approach, where the delivery of data is analogous to multicasting [83]. ICN permits the network to provide content from any source where it is placed (including in-network cache), and satisfies the user requirements to send and receive data, regardless of its location. One of the most prevalent approaches of ICN is Named Data Networking (NDN) and is based on a query-response model

[91].

2.2.1.1 Off-Path Caching

This approach is similar to CDN server placement or conventional proxy caching. In this method, the common issue is cache placement, i.e., making decision of what to cache and where. When old items are discarded or new items are cached, then informing the name resolution system (NRS) can increase the overhead which is a major problem in off-path caching [31].

In the COMET and DONA, cached information are not advertised upward in the autonomous system (AS), but can only be announced within the AS. The name prefix tables should be updated in the coupled version of SAIL, CCN, and CONVERGENCE, yet it is not clear how this can be accomplished as the proposed routing protocols for publicizing name prefixes follow the mechanism of flooding. In the decoupled version of SAIL and PURSUIT, stored data must be reported inside the local Distributed Hash Table (DHT) of an AS. The MobilityFirst depends on a worldwide query method for name resolution, therefore, it additionally confronts issue here, i.e., it is not clear how those duplicates which are locally stored can be advertised just inside an AS [31].

2.2.1.2 On-Path Caching

This approach is opportunistic, i.e., network nodes cache packets that pass through them. On-path caching is uncomplicated approach to content caching as they move from sender to receiver. In this method, when a packet arrives, the router answers with a copy locally cached without contacting the NRS [1].

As contents are stored within the network in on-path caching, the communication and computational overhead is reduced, but the chances of cache hit might also be minimized. A critical issue in this approach is the decision of a router to place contents for improving the cache hit ratio. Although the principle of on-path caching is natively supported by all ICN architectures, when content routing and name resolution are decoupled, there are less chances to take advantage of opportunistic caching, as data path and name resolution paths are generally different from each other. As the contents on the data path can be opportunistically stored, the next request for the same content follows a different name resolution path, which can also reduce the possibility of a cache hit [31, 1].

Similarly, as contents are placed on the data routing path when the data routing and name resolution are coupled, a cache hit occurs if the next request arrives on the same name resolution path [1]. A detailed explanation about off-path caching and on-path caching can be found in [92, 93, 94, 95].

2.3 Existing Cache Management Strategies

Caching multimedia applications is a great concern of the ICN because the ICN cache strategy is still in initial stage and has a lot of space for the research community. Numerous cache management and replacement strategies have been developed, which are presented in Figure 2.2 and explained in the following subsections.

2.3.1 Hash-routing

Hash-routing [96] schemes are proposed for off-path caching. According to hash-routing schemes, if a content request arrives at an edge router, it calculates a hash function that maps the content identifier to a certain caching router and forwards the request to that particular router. The router then returns the requested content to the user if it is maintained in its cache, otherwise forwards it to the original source.

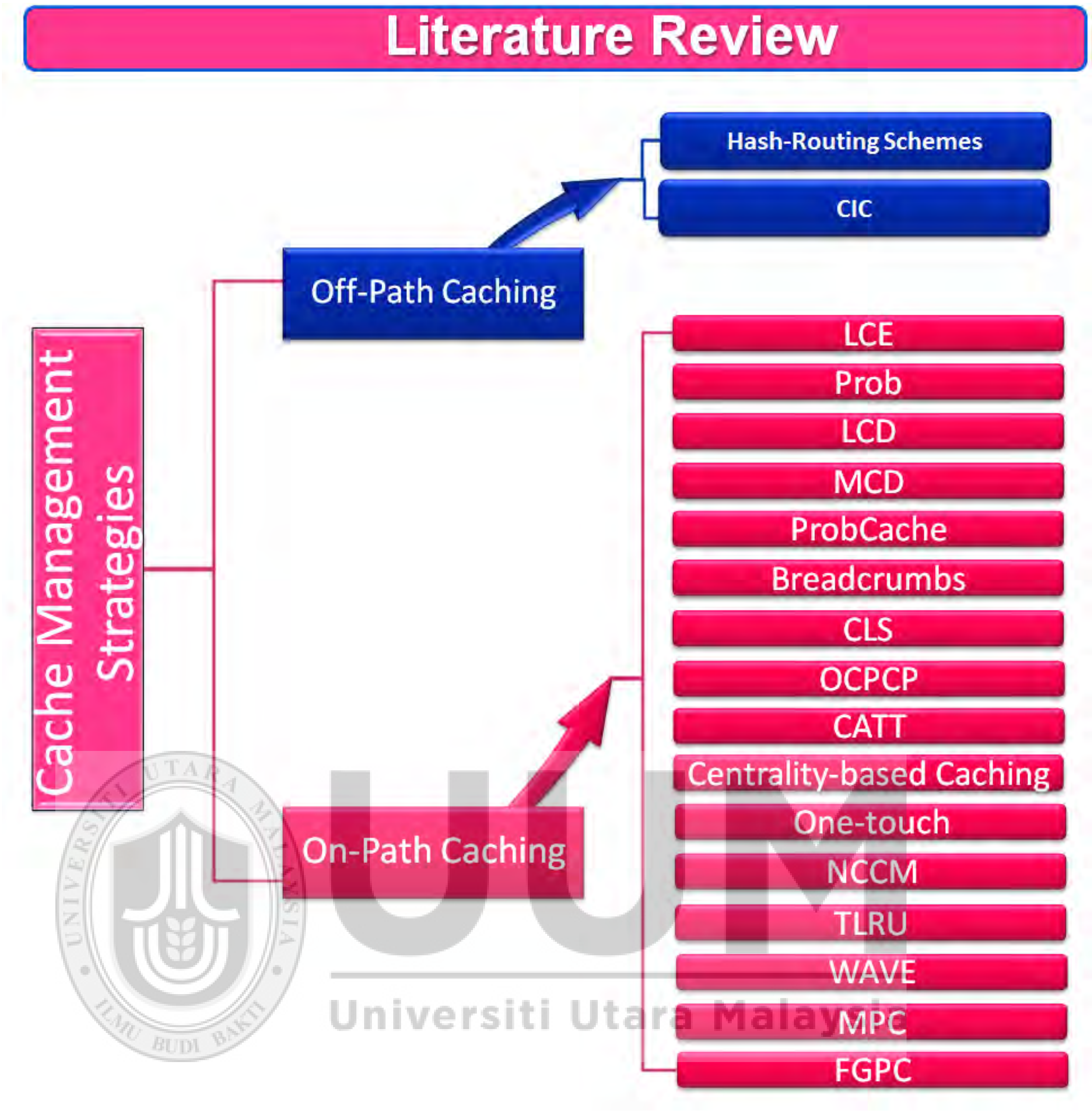


Figure 2.2. Caching Strategies

Similarly, when a router forwards the requested content to the user, only that router can cache it which is associated to the content identifier. The main objective of the hash-routing schemes is to remove caching redundancy as the hash function indicates to cache the content where it is found.

If the content item (entering the domain through the edge router ER-3) is forwarded to router ER-2 and the specified cache is at node N-4, as shown in Figure 2.3, then multicast delivery becomes more effective. Likewise, a symmetric delivery is preferred if the specified cache is at node N-3. In this case both approaches, i.e., multicast

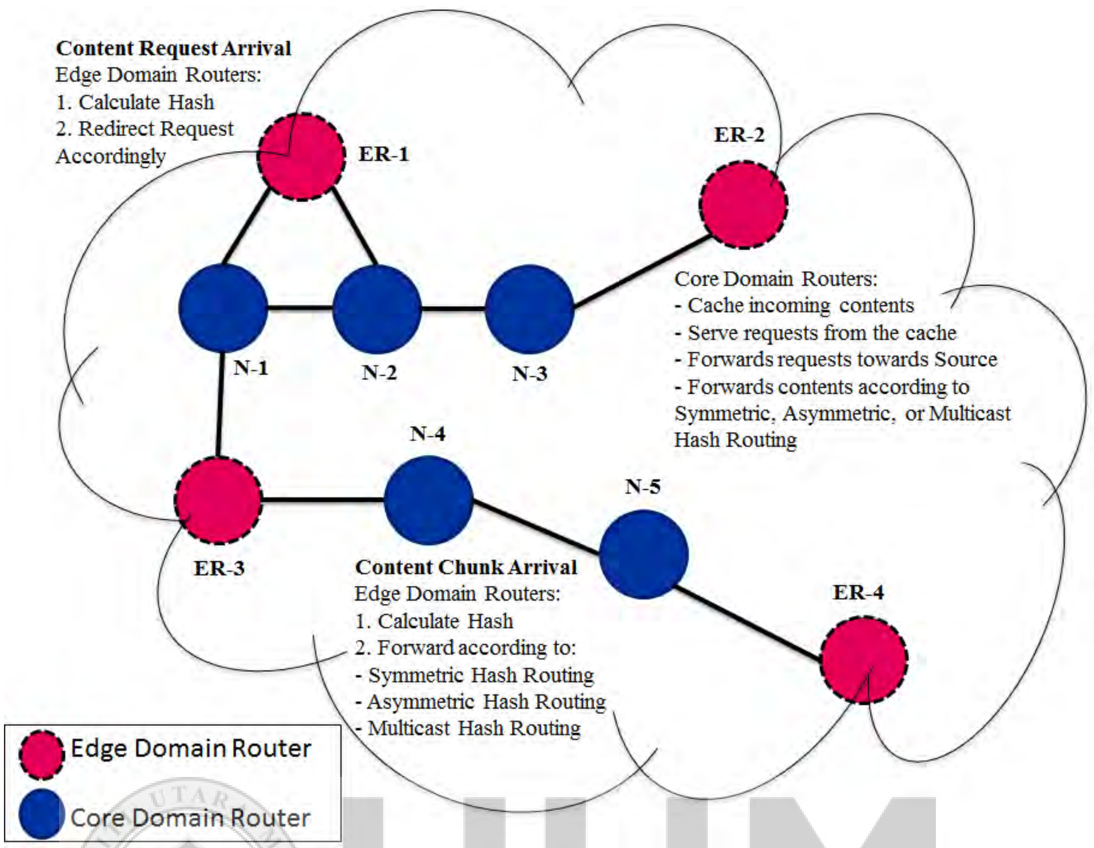


Figure 2.3. Hash Domain Routing Functional Architecture [96]

hash-routing and symmetric, can achieve higher cache hits. However, in both of these schemes the possibility of intra-domain link load will be potentially high.

2.3.2 Cooperative In-network Caching (CIC)

This is another off-path caching strategy. In cooperative In-network Caching (CIC) [97], a content is divided into different chunks and cached at more than one node such as router. For instance, a content is 12 chunks: each chunk is of the same size, and the cache capacity of a router is only to store six chunks, then this content is divided into two same size chunks and cached at two different routers. To better understand the concept of CIC, consider Figure 2.4, where each route has a cache capacity of 7 chunks, but the chunks are cached based on *modulo 3* division. In other words, a router does not cache all chunks of a content in a sequence, but a part of those chunks are stored according to *modulo k*. Let us say that $k = 3$, then router C will cache

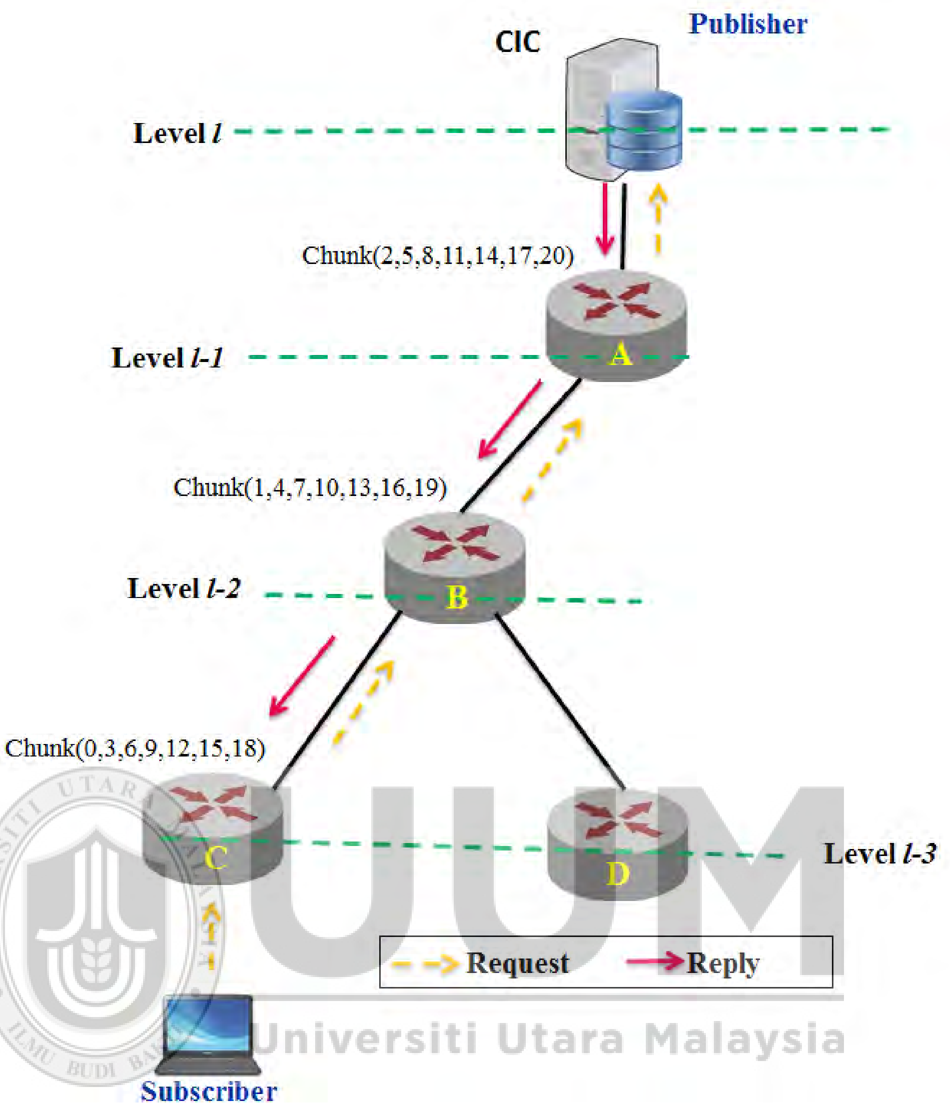


Figure 2.4. CIC Topology

chunks 0,3,6,9,12,15,18 and router B and A will cache chunks 1,4,7,10,13,16,19 and 2,5,8,11,14,17,20, respectively.

Now, if the subscriber requests a content whose part is chunk 15 – cached at router C, it will not be forwarded to the server, but router C will satisfy it locally. Similarly, if a next request arrives for another content whose part is chunk 5 – cached at router A, router C cannot satisfy it, therefore, it will be forwarded to router A. Hence, both the requests are satisfied by the local machines because of this cooperation. The CIC is an off-path caching strategy because the contents are cached at the routers which are not available on the publisher-subscriber path.

The caching diversity which is the amount of different chunks in the cache can be improved by CIC with avoiding caching redundant chunks. Moreover, the average response time: the turnaround time between the request initiation and the receiving of the requested chunk, can also be enhanced. However, the CIC adds two extra tables, named Collaborative Router Table (CRT) and Collaborative Content Store (CCS). CRT is used to keep track records of all cached chunks along with their IDs, whereas CCS is used for keeping the sequence numbers and names of all the cached chunks at every router. These operations involve some extra computational overhead and therefore increase the searching time in case some chunks need to be evicted for caching a new arrived content.

2.3.3 Leave Copy Everywhere (LCE)

This strategy was designed for multi-level caches. In Figure 2.5, the user (subscriber) requests a content which is available on the server (publisher). The request is forwarded through router C,B,A to reach the publisher. In LCE [30, 32], when a hit occurs either at the main server or a level l cache, a copy of the requested content is stored in all intermediate nodes (levels $l - 1, ..., 1$) on the route from the node on which hit occurred through the requesting user.

The LCE caches contents at all intermediate nodes, therefore, if a new content request arrives, it is not forwarded to the server but the intermediate node, which has a copy of that content, replies with data. Through caching a content at each router the bandwidth utilization can be improved effectively because the requests are not forwarded to the server but satisfied locally. However, due to redundant replicas of the same content, the cache of all intermediate nodes will become full, and therefore new arrived contents

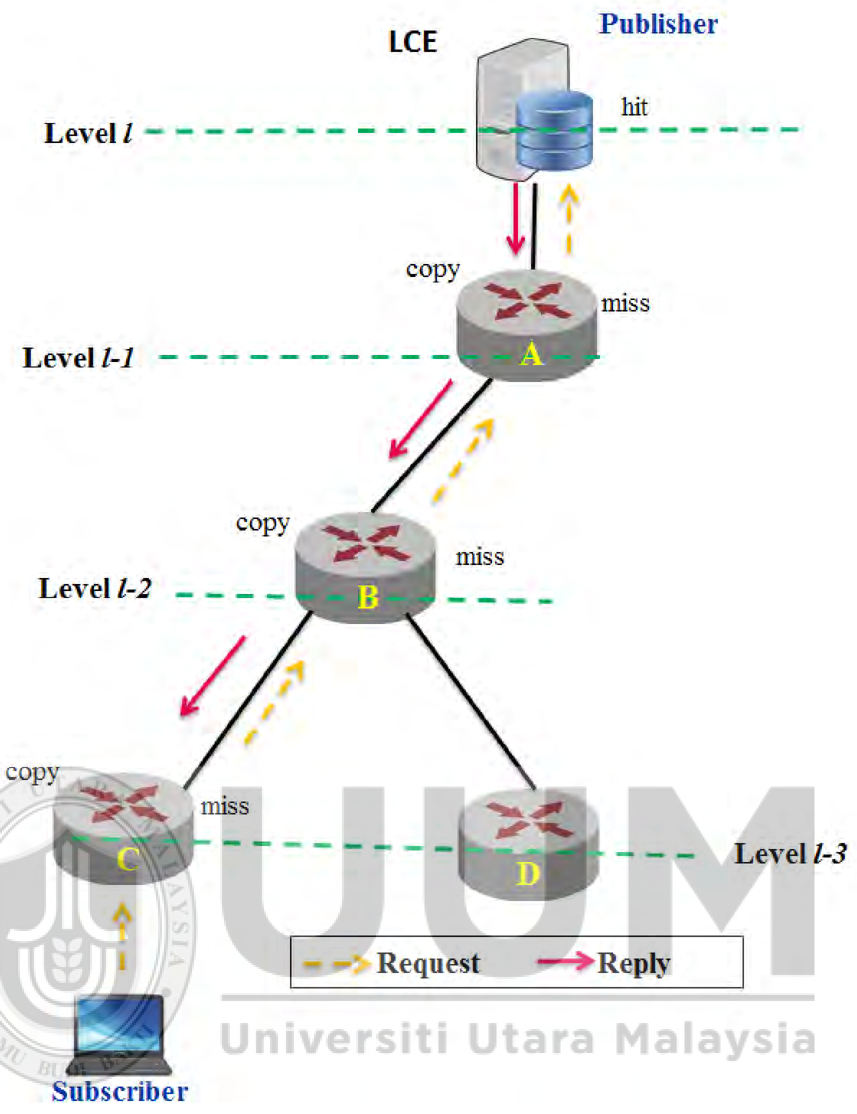


Figure 2.5. Operation of LCE

will never contribute in the cache hit.

It seems that caching content at all nodes will increase cache hit ratio, but this phenomenon is impractical because when the cache of a node overwhelms and a new content arrives then it needs to replace one of the cached contents. Most of the strategies follow Least Recently Used (LRU) or Least Frequently Used (LFU) policy for content replacement [98], therefore, the evicted content may be a popular one, which is possible in both LFU and LRU replacement. Now, in case of a new request arrival for that evicted content, it will not find the content locally, therefore, the request will be

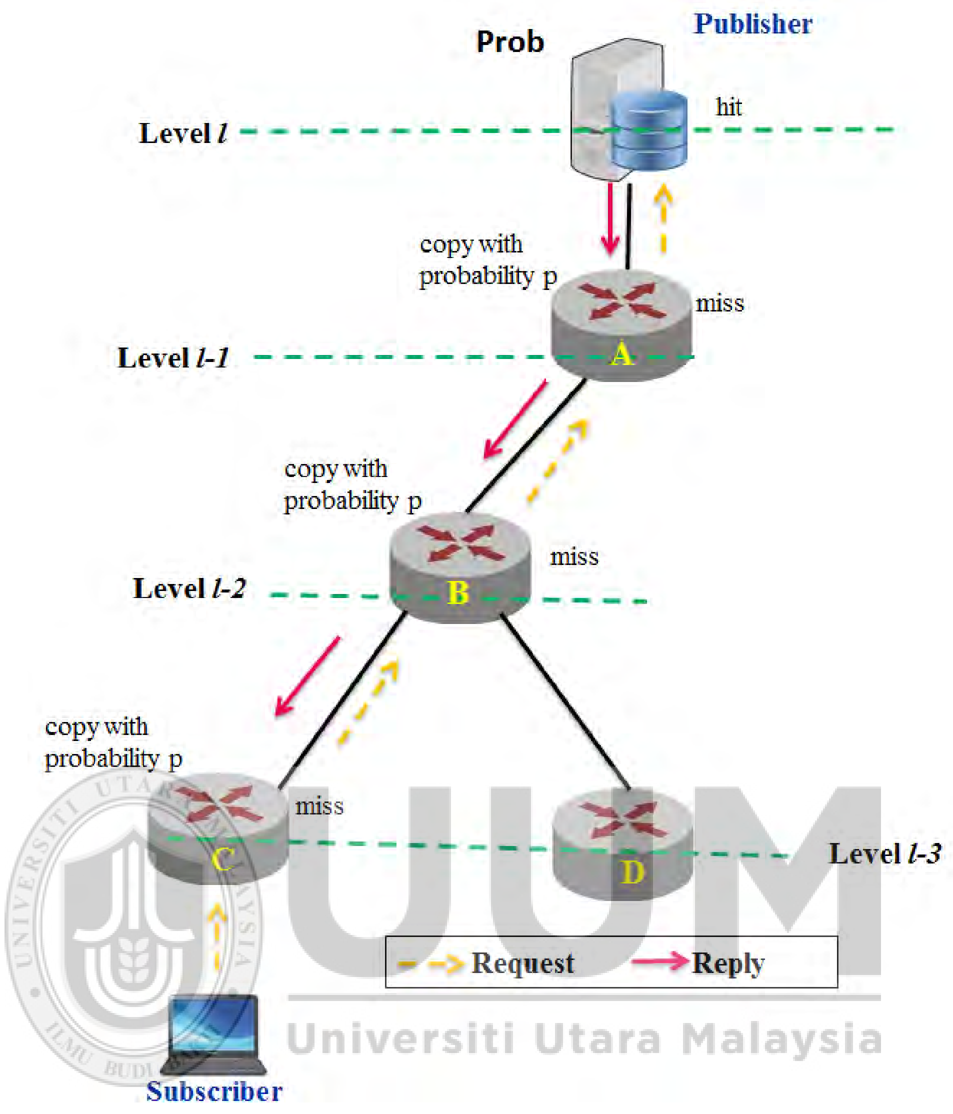


Figure 2.6. Operation of Prob

forwarded to the server. Consequently, it will affect the overall network performance with respect to cache hit rate, content retrieval delay, and bandwidth consumption.

2.3.4 Prob

Prob [32] is a randomized form of LCE, where each node can cache a copy of the requested content on the path from the server or node where hit occurred to the requesting host (see Figure 2.6). A node in the middle of the network does not cache a copy with probability *1-p*, but can keep only a local copy with probability *p*. The operation of Prob(*1*) is similar to that of the LCE strategy.

To know the concept of probability *p* and probability *1-p*, let us consider the following example [99]:

A Bernoulli trial, which is a random experiment, has two possibilities, i.e., success or failure.

Accessing a content and considering *hit* as success and *miss* as failure.

Let X be a Bernoulli random variable that takes either 0 or 1 as a value and

$$P(X = 1) = p \tag{2.1}$$

and

$$P(X = 0) = 1 - p. \tag{2.2}$$

It is easy to check that the mean and variance of a Bernoulli random variable are

$$E(X) = p \tag{2.3}$$

and

$$V(X) = p(1-p). \tag{2.4}$$

Consider an experiment where a content is requested *n* times; each request has two possible results labeled as *hit* or *miss*. The requests are independent, i.e., the result of any request has no impact on the probability of the others. The probability of success in each request is constant and is denoted by *p*.

Let *X* be a binomial random variable having *n* and *p* parameters, then the probability mass function is

$$f(k) = P(X = k)\binom{n}{k}p^k(1-p)^{n-k}, \forall k = 0, 1, 2, \cdots, n. \tag{2.5}$$

$\binom{n}{k}$ calculates the number of results which include exactly $n-k$ misses and k hits .

Example:

A content is requested 6 times. The probability of hits on any request is 0.3.

Let *X* denotes the amount of hits occurred then calculate:

$P(X = 2)$.

If a success is called *hit* then this *X* has a binomial distribution with $p = 0.3$ and $n = 6$.

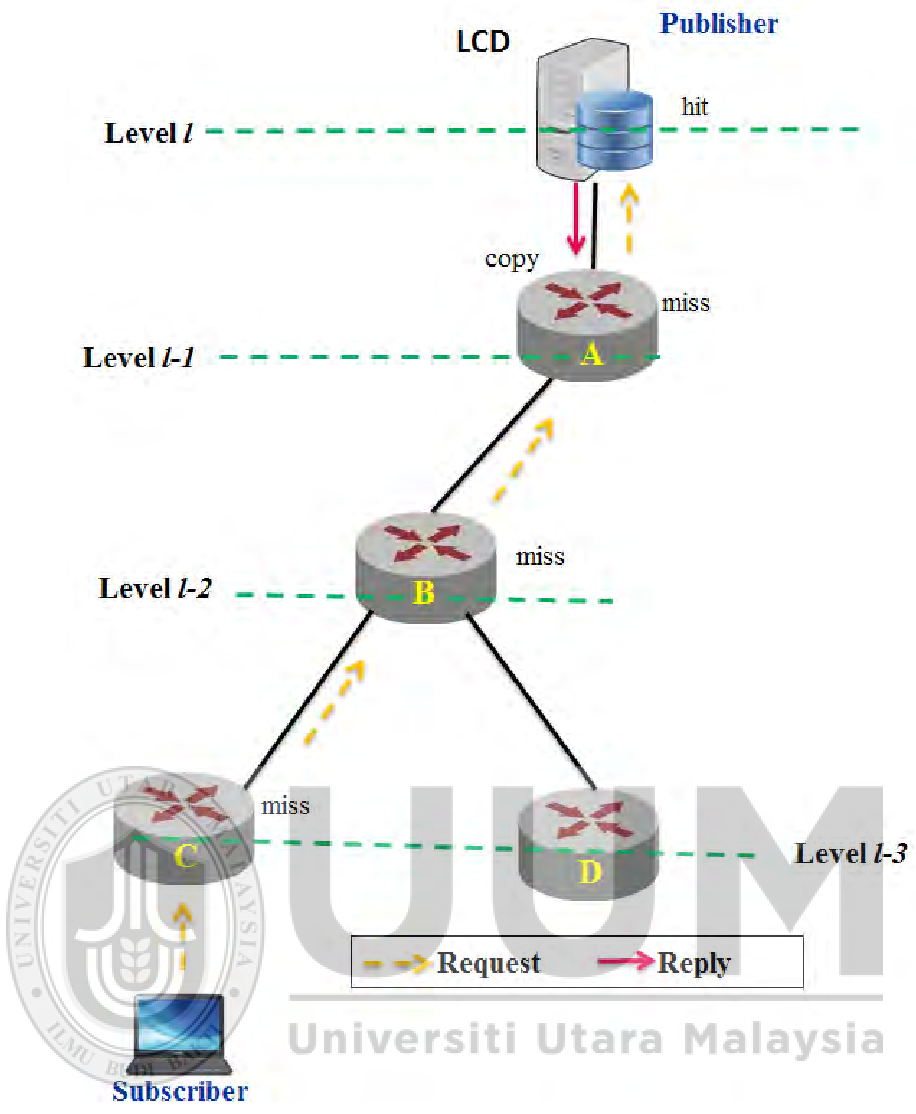


Figure 2.7. Operation of LCD

Put in Equation 2.5, we get

$$P(X = 2) = \binom{6}{2}(0.3)^2(0.7)^4 = 0.324135. \tag{2.6}$$

2.3.5 Leave Copy Down (LCD)

In LCD [32], a copy of the requested content is stored merely at the (l − 1) level node, i.e., the one which is located instantly down the node where hit occurred, as shown in

Figure 2.7. To forward a content to the leaf node, unlike LCE, LCD requires multiple requests, with each request forwarding a fresh copy of the content one hop closer to the client.

With caching contents only at one node, i.e., $(l - 1)$ level node, LCD increases the cache hit ratio and effectively utilizes the cache space. However, if some contents are requested frequently (popular contents), then at some time all nodes on the link will cache these contents and hence, cache of the nodes will become overwhelmed and the space will not be available for the new arrived contents. The LCD also uses LRU for content replacement in case the cache becomes full, therefore, excessive eviction operations happen and the new arriving contents replace the popular ones. Consequently, all new requests for those evicted contents are forwarded to the server and in turn a maximum utilization of bandwidth is experienced.

2.3.6 Move Copy Down (MCD)

The functionality of MCD [100] is analogous to LCD, but the only difference is that a hit at level *l* shifts the requested content to the underlying cache. In Figure 2.8, as hit occurs at level *l-1,* the content is moved to level *l-2*. In this approach, no deletion is required when a content is hit at the main server, but the deletion takes place only if the requested content is hit at a location rather than the original server. The main objective of MCD is to minimize the content redundancy between the server and the requesting host.

Looking at Figure 2.8, the subscriber requests a content which had already been accessed by some user(s) and cached at router A. As the request reaches router A, the hit occurs there and router A replies with that content. Here, unlike LCD, the copy of that content is moved to the down node, which is router B, and the cached copy at router

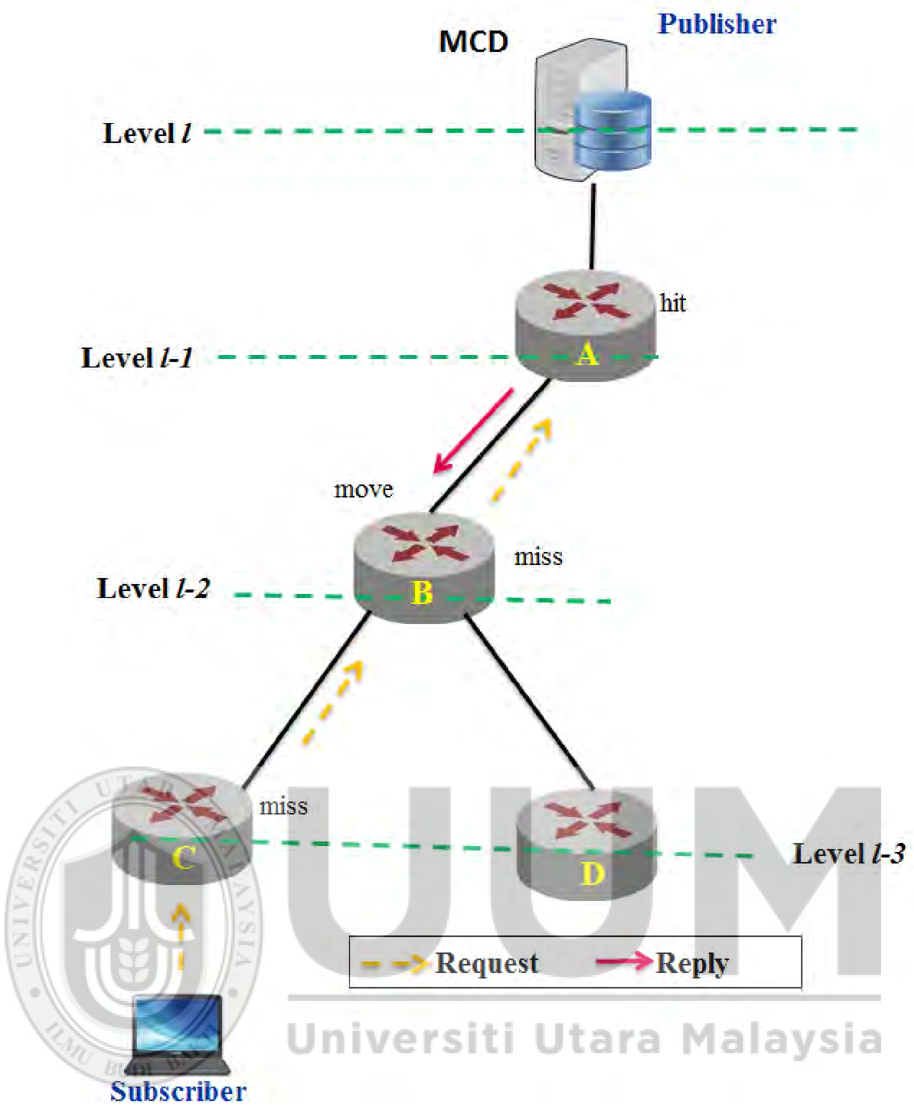


Figure 2.8. Operation of MCD

A is deleted. Furthermore, if the same content is requested again by this subscriber, its copy at router B is evicted and moved down to router C. This process efficiently utilizes memory as well as decreases the stretch. However, if another subscriber - connected to router D, requests the same content, the request reaches router B via router D, where no copy of that content is available. Therefore, the request is sent to the publisher and consequently the average cache hit rate is reduced and content retrieval delay is considerably increased.

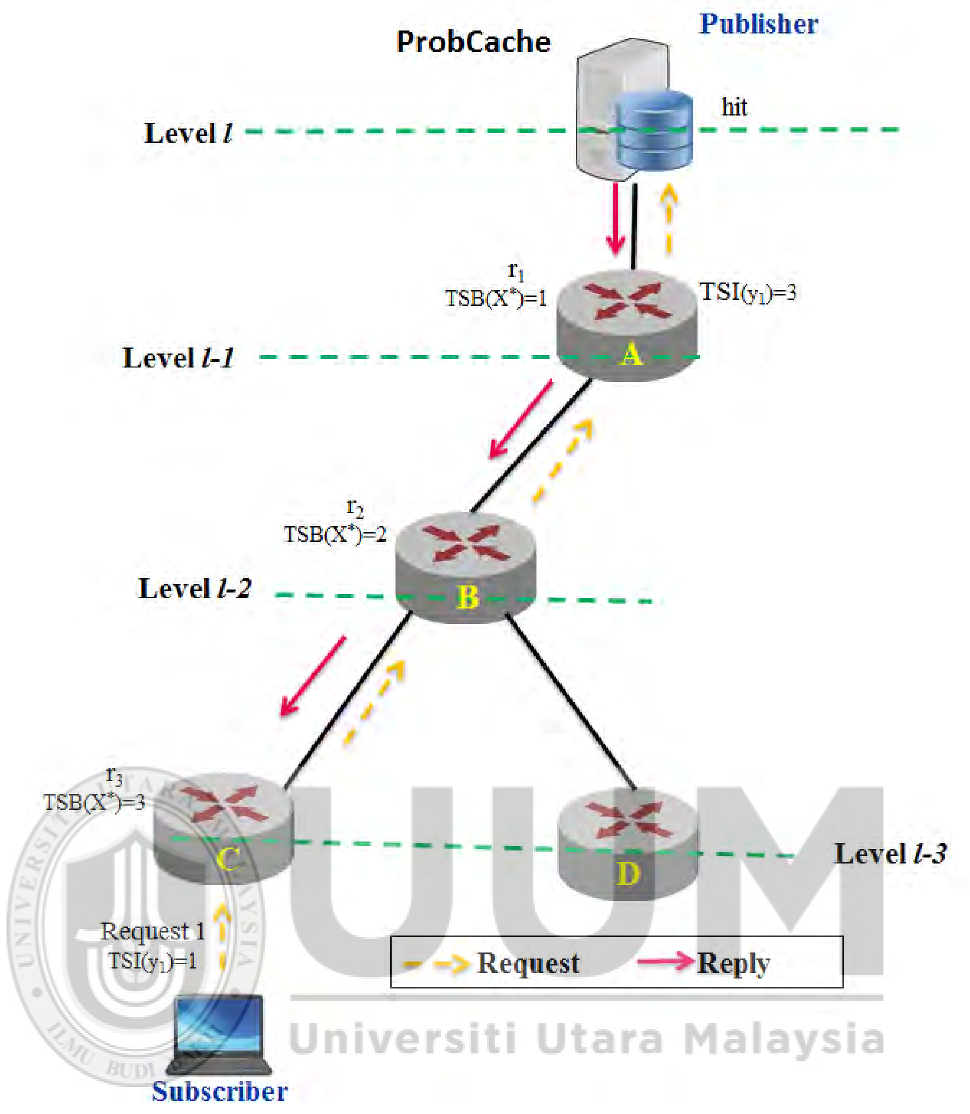


Figure 2.9. Design Topology of Probabilistic Caching

2.3.7 Probabilistic Caching (ProbCache)

In ProbCache [41], as shown in Figure 2.9, the *Time Since Inception* (TSI) is a field required in the ICN *request message* headers, while the *Time Since Birth* (TSB) is an important filed in the *content message* headers. Where r represents router, n is the number of caches on the path, and X is the hop-distance from server. In the given topology, a user is connected to the server at a distance of four hops. When a cache hit occurs, the TSB value of the content packet is set to 1, while the TSI value is replaced by that of the *Request* packet.

This policy is considered as a *resource management* approach to in-network caching. ProbCache attempts to maximize the amount of different content items stored along a communication route. In other words, it tries to reduce redundancy between network caches. Thus, the probability of finding content along the path becomes higher for subsequent requests. But, the *Hop decrement* (the percentage of *content request hops* and the *path hops* between the user and server), which is an important parameter for expediting the process of content access, will become lower in ProbCache. The lower the *Hop decrement* the higher the content delay, and vice versa. The *Hop decrement* in ProbCache decreases because it frequently replaces contents at nodes near the user(s).

2.3.8 Breadcrumbs

Breadcrumbs [101] is a transparent and best-effort simple searching and content caching technique. Each breadcrumb is a $5-tuple$ entry, indexed by a global content ID. This ID holds a set of information, i.e., 1) the ID of node to which the content was pushed down, 2) the ID of node from which the content was received, 3) the most recent time of the requested content, and 4) the most recent time of the forwarded content (see Figure 2.10). As the content is downloaded, a trail is created and maintained at each router for caching the routing history. This trail will route the subsequent request downstream for the same content towards the router's directory, instead of upstream towards the server. Similarly, to check whether searching downstream or not, a time threshold is used in this scheme. One of the main advantages of a downstream request routing technique is to minimize the server overhead. Another benefit of breadcrumbs is to reduce the content download time if it is found downstream.

However, if the content is evicted at each router, the request may suffer missing downstreams, and hence the overall content delay, during downloading, will undoubtedly increase.

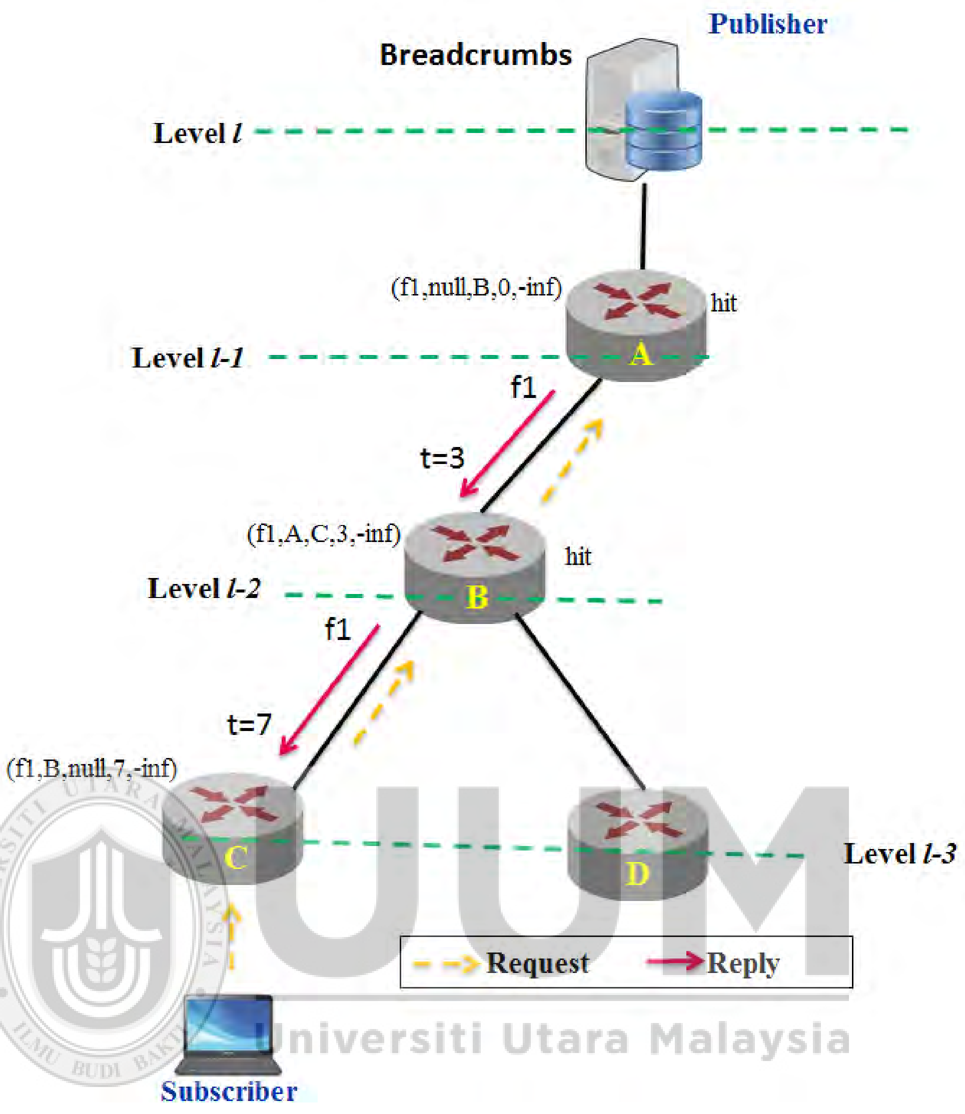


Figure 2.10. Breadcrumbs Topology

2.3.9 Chunk Caching Location and Searching Scheme (CLS)

The main objective of CLS [102] is that a hit at level *l*, as in MCD, drags the requested chunk down to the *l-1* level node (see Figure 2.11). When the content is downloaded, as in Breadcrumbs, every router creates a trail for caching the routing history. Each chunk is a *4 – tuple* entry, indexed by a global chunk ID. This chunk ID holds a set of information, i.e., 1) the ID of a node from which the chunk was received, 2) the ID of a node to which the chunk was pushed down, and 3) the number of hops between this router and server. Each router adapts the PIT to cache the trail information of formerly stored chunks. The major difference between Breadcrumbs and CLS trails is

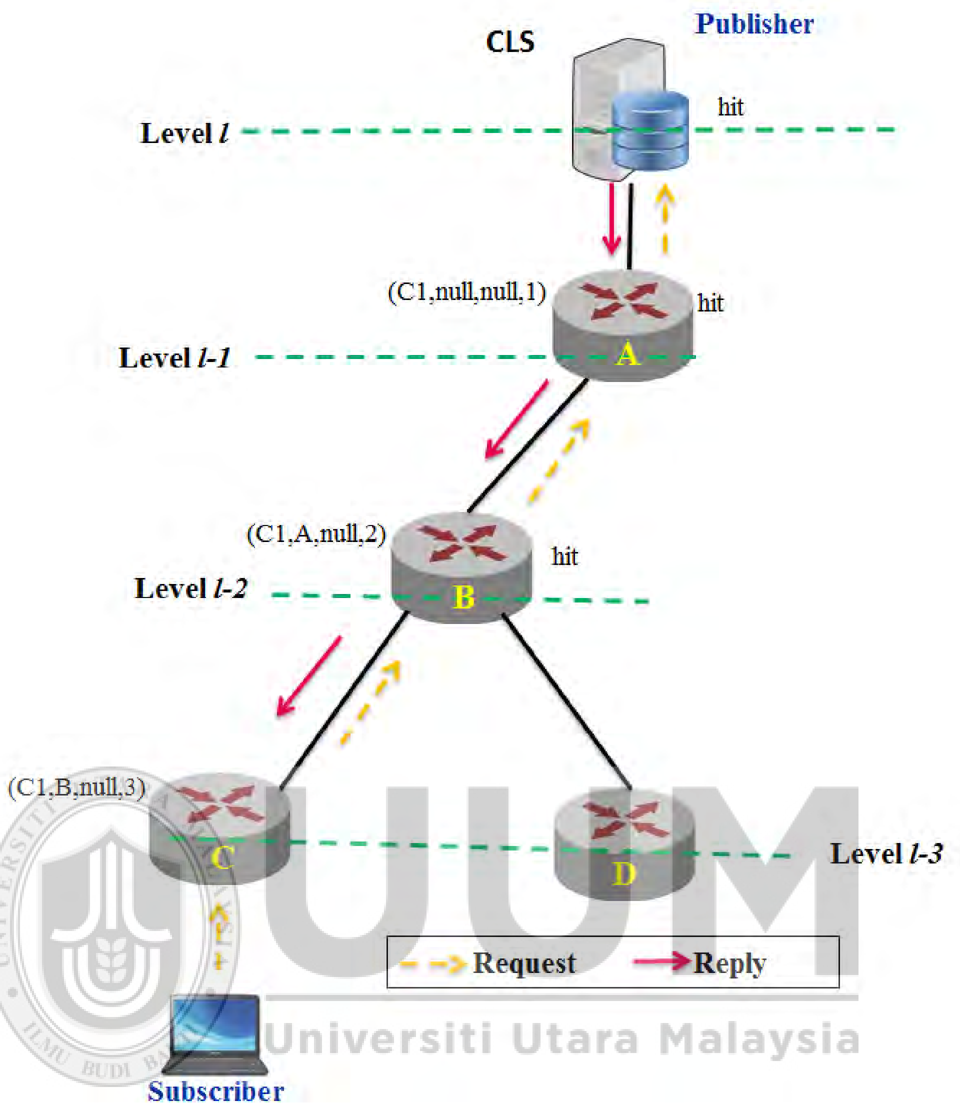


Figure 2.11. Operation of the CLS Scheme

that the Breadcrumbs trail is generated at the time the content is pushed down, while in CLS the trail is maintained at the time of chunk caching. This guarantees that the CLS saves cache space as it deletes the trail on time and consequently finds the chunk if search downstream.

However, according to the CLS trail, an intermediate node cannot judge whether the server is close or the cache copy. Therefore, it cannot ensure the placement of chunk at the nearest node to the user.

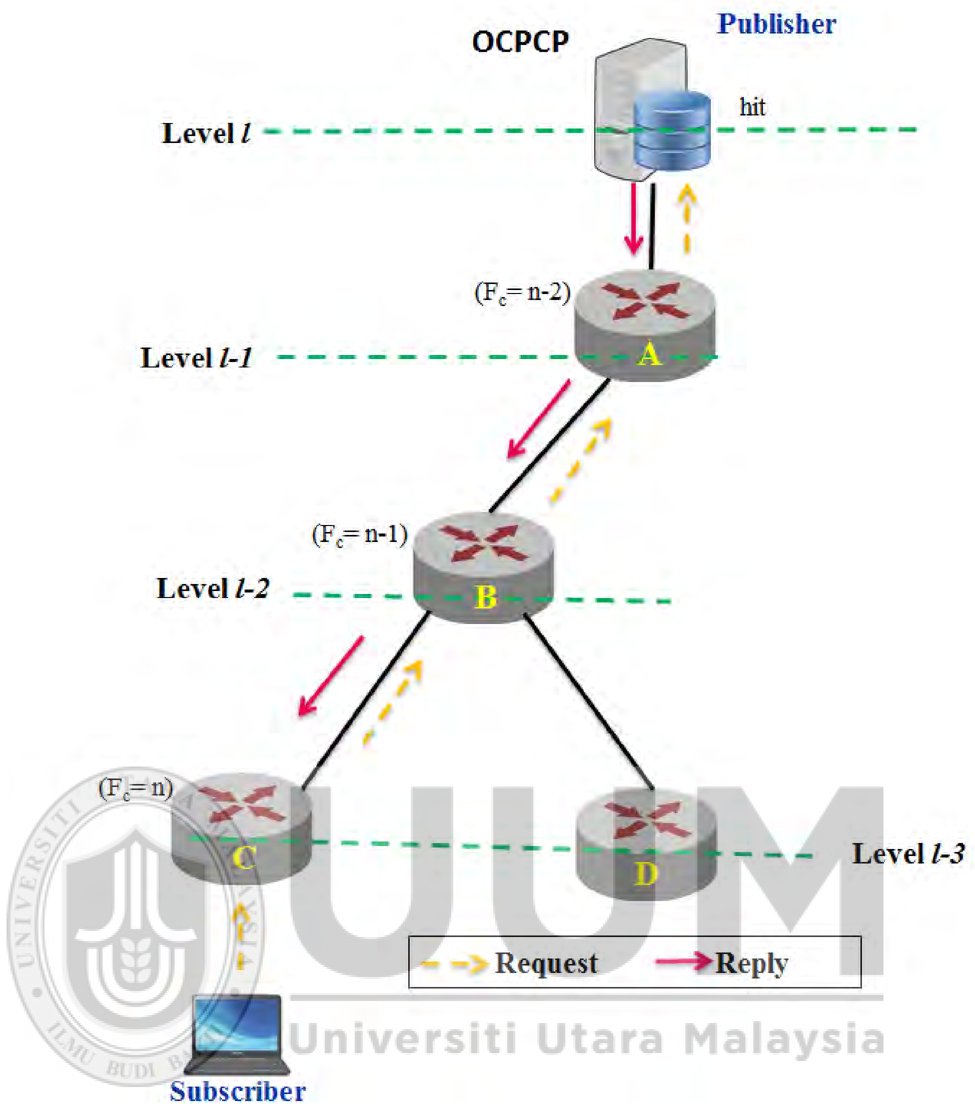


Figure 2.12. An OCPCP Topology

2.3.10 Optimal Cache Placement based on Content Popularity (OCPCP)

In OCPCP strategy [103], the popularity of incoming content is calculated on the basis of cached contents and the new content is cached based on its popularity value. The OCPCP takes a caching decision by just considering the records of content requests at a single node. That is, the more frequency of requests for a content, the higher probability of next requests. In other words, if the amount of content requests is high, it has a higher popularity and is therefore considered for caching.

Consider Figure 2.12, initially the cache of all routers is empty. The subscriber requests a content which is available on the publisher. The request reaches the publisher via routers C, B, and A. As the popularity of a content (which is the number of requests) increases, it is cached on the router near the subscriber, i.e., router C in the given figure. When the network gains stability, router C caches the most popular contents, denoted by $F_c = n$, where F_c is the frequency of content requests. Routers B and A would cache the second and third highest popular content, respectively. The OCPCP counts this frequency of content requests through a data structure called Request Record Table (RRT). This calculation is based on the following equation:

$$F_c = \frac{\sum_{i=1}^{n} \delta_c(i,j)}{N} \tag{2.7}$$

where F_c is the total frequency of contents; N is the total number of requests; $\delta_c(i,j)$ is a function if the new content c_i and the j^{th} content of RRT are equal then its value becomes 1.

Content placement in OCPCP is based on the following Content Placement Algorithm (CPA):

Step 1: Store the incoming content if the cache has capacity, otherwise, calculate the popularity of F_{ci}.

Step 2: Check whether F_{ci} has the highest popularity, i.e., $F_{ci} = n$, if yes, follow Step 3, otherwise go to Step 4.

Step 3: Cache content F_{ci} on LRU basis.

Step 4: Update the value of content c_i in RRT.

The OCPCP caches contents only at a router located near the subscribers, therefore, the content redundancy and the retrieval delay are minimized. This placement of contents near the subscriber(s) can also improve the network throughput in terms of bandwidth utilization as the requests can be satisfied locally instead of forwarding to the publisher. However, because of the rapid increase in the number of Internet users and demand for multimedia applications, the chances of filling router's cache are very high. As a result, the new arriving contents may replace the cached popular ones and therefore they will no longer participate in the cache hit ratio.

2.3.11 Cache Aware Target Identification (CATT)

In CATT scheme [104, 105], the content is cached at a single node on the path from the publisher to the subscriber. CATT introduces the expected quality of contents to help a router forward content requests. If a content is cached at a router, its expected quality is supposed to be E_q, while the expected quality for a non-cached content, denoted by $E(NC)$ is calculated as:

$$E(NC) = \frac{E_q}{R_h + 1} \tag{2.8}$$

where R_h is the number of hops between the caching router and the current one.

If a router caches a content, it multicasts the content related information to its neighbors. As a neighbor router receives this information, it first calculates the previous expected quality of the content and then sets its new expected quality.

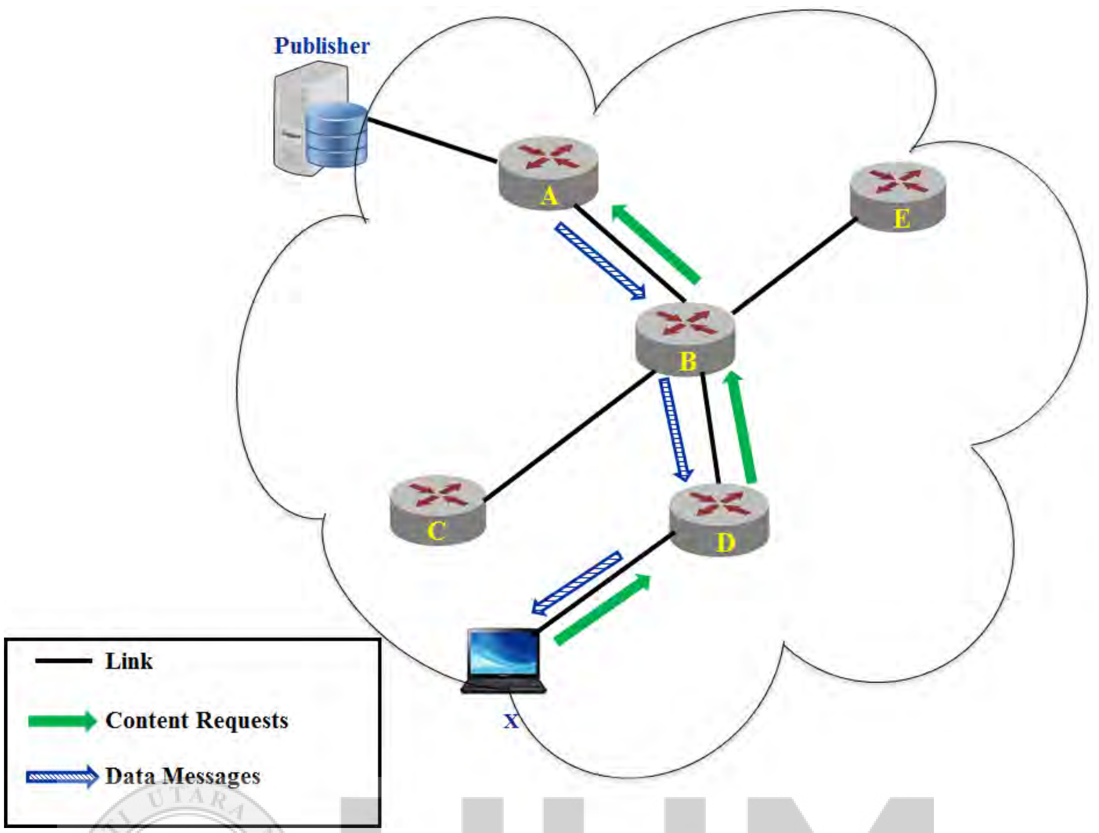


Figure 2.13. A CATT Topology

In this scheme, on the edges of an AS some CATT nodes (CATNs) are deployed as caching nodes. CATNs have three major modules, namely cache, repository, and routing. The provision of caching capabilities for all CATNs is responsibility of the first module, i.e., cache. A caching decision is made when a content is downloaded from the publisher to the subscriber. The second module, repository, manages the cache where a subscriber publishes contents. The last module, routing, forwards the content requests based on their names. In CATT scheme, instead of FIB, the Potential Based Routing (PBR) is used for content retrieval. If a node receives a content request and it is not stored in its cache, it compares the expected quality of that content with its neighbor's PBR and forwards the request to that neighbor which has the highest expected quality for this content.

In Figure 2.13, router B receives a content request from user X via router D and finds that its highest expected quality is at router A. Therefore, it forwards the request to router A rather than router C or E.

As CATT caches contents at the edges of AS, it reduces content redundancy and retrieval delay [63]. However, if the cache of edge routers overflow then the new contents will replace the stored ones, therefore, the increase in cache hit ratio cannot be guaranteed.

2.3.12 Betweenness-Centrality

In Betweenness-Centrality [106], the content items are stored only once along the path, particularly at the router having the highest betweenness centrality, i.e., the router having the maximum number of shortest routes traversing it. Consider the example in Figure 2.14, a user X requests a content which is available on the publisher. The publisher will reply with the requested content and copy of it will be cached at router B which has the highest betweenness centrality. Furthermore, if user Y and/or Z request the same content, router B will satisfy their requests locally rather than forwarding to the publisher. In case more than one router has the highest betweenness centrality, then the content is cached in a router located near the customer. In this strategy, caching decisions are made on the basis of betweenness centrality, but it is challenging for a router to find the value of its betweenness centrality if it is deployed on every mobile ad-hoc network. In addition, if the cache of router B becomes full and a new content needs to be cached, the LRU policy will replace one of the cached contents, which may also be popular, and the subsequent request(s) for that content will be forwarded to the publisher. This operation can reduce the cache hit rate as well as maximize the bandwidth utilization.

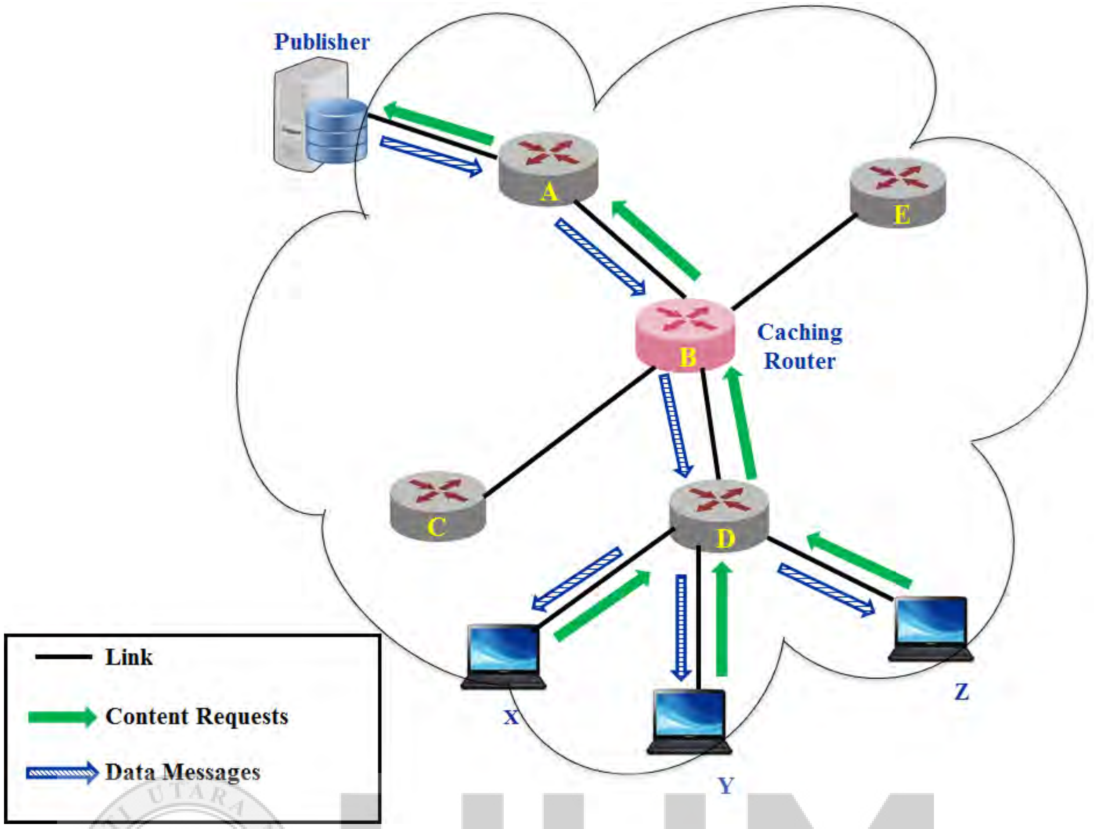
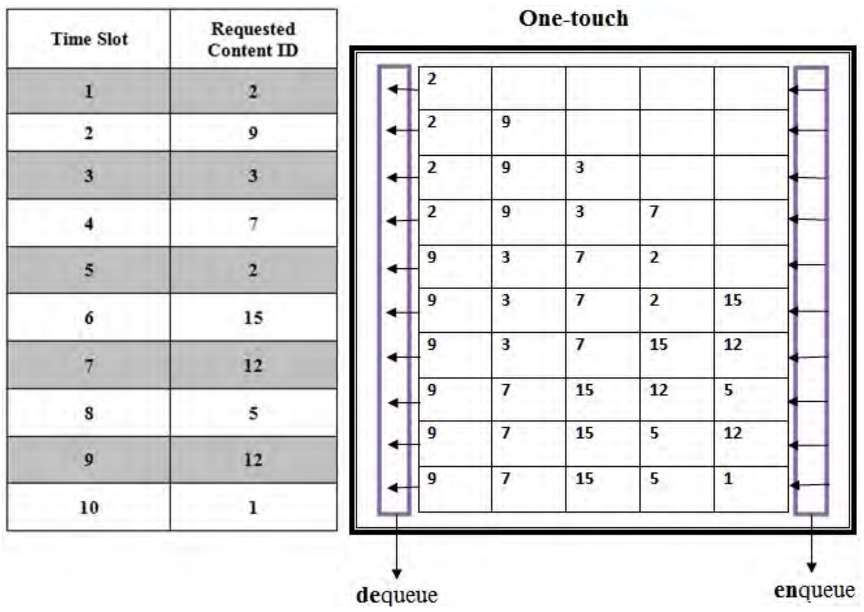


Figure 2.14. An Example Topology with Caching Location at Router B

2.3.13 One-touch Caching

In one-touch caching scheme [107], the most recently requested content is preserved in the server and existing content is randomly evicted if the cache space is full. If free cache space is available in the server, all the requested contents are accumulated there. The requested content is replied from the cache server if it is maintained in its cache.

The content management approach for one-touch caching scheme is shown in Figure 2.15. In the given example, the server's cache can accommodate only six contents. Until the last time slot, which is 6th in the given figure, no deletion took place as the server had storage capacity. When a content is requested, it is replied from the server if available, as presented in the 5th time slot. However, if a new request arrives after the 6th time slot, one of the stored contents needs to be evicted from the server's cache. This content eviction takes place in a random fashion. It is shown in the 7th time slot



Time Slot	Requested Content ID
1	2
2	9
3	3
4	7
5	2
6	15
7	12
8	5
9	12
10	1

2				
2	9			
2	9	3		
2	9	3	7	
9	3	7	2	
9	3	7	2	15
9	3	7	15	12
9	7	15	12	5
9	7	15	5	12
9	7	15	5	1

Figure 2.15. One-touch Caching Strategy [107]

that a new arrived content, whose ID is 12, is cached and the stored content, whose ID is 2, is evicted. This eviction of content with ID 2 is in a random way.

Due to the simplicity of one-touch caching scheme, there is no need for sorting operation, tedious caching, and replacement processing. Although, one-touch caching has a little overhead compared to LRU and LFU, but as the contents are not cached locally in this scheme, all requests are served by content servers and thus maximum bandwidth may be utilized during request to and/or reply from the server(s). As a result, unavoidable content delay may occur.

2.3.14 Network Coding based Cache Management (NCCM)

The NCCM scheme [108] is developed to achieve near-optimal caching and routing solutions for ICN. Here, a novel framework based on Software Defined Network (SDN) is proposed for cache management in ICN which jointly considers content routing and caching strategy through Linear Network Coding (LNC). Using this scheme, the problem of minimizing the network bandwidth cost is formulated through LNC. The linear combination of the original data chunks can be cached by the three routers, i.e.,

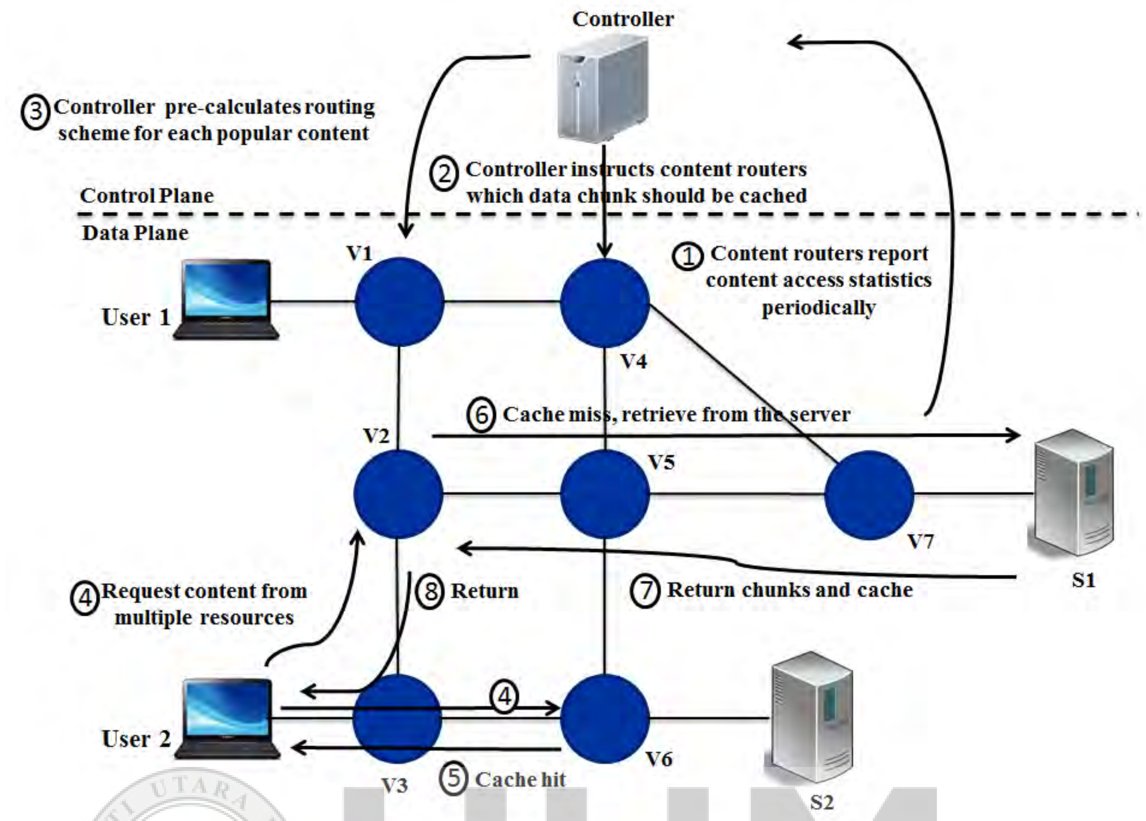

Controller
3 Controller pre-calculates routing scheme for each popular content
2 Controller instructs content routers which data chunk should be cached
Control Plane
Data Plane
1 Content routers report content access statistics periodically
User 1
V1
V4
V2
V5
V7
6 Cache miss, retrieve from the server
S1
4 Request content from multiple resources
8 Return
7 Return chunks and cache
User 2
4
V3
5 Cache hit
V6
S2

Figure 2.16. A Cache Management Framework [108]

V1, V4, and V6 in Figure 2.16. In order to obtain the best possible solution, content request statistics should be reported periodically with a fixed time period by all CRs. For popular content routing, the controller needs to discover the possible path and therefore configure flow tables in CRs on the path. When a request arrives at CR from its local end user, it will first check if the same content with the same sequence number is already requested and the data chunk is not returned, then it will not send this request. If not, the request will be forwarded through an outgoing interface when the number of data chunks that can be received from this interface is greater than the number of data chunks received from this outgoing interface. A CR will create a new coded data chunk (if it receives a cache hit) by randomly combining the cached data chunks and return it. Similarly, the CR will obtain the requested data chunks from one of the servers if it misses a cache. As the CR obtains the coded data chunks, it caches it and forwards to the requested user.

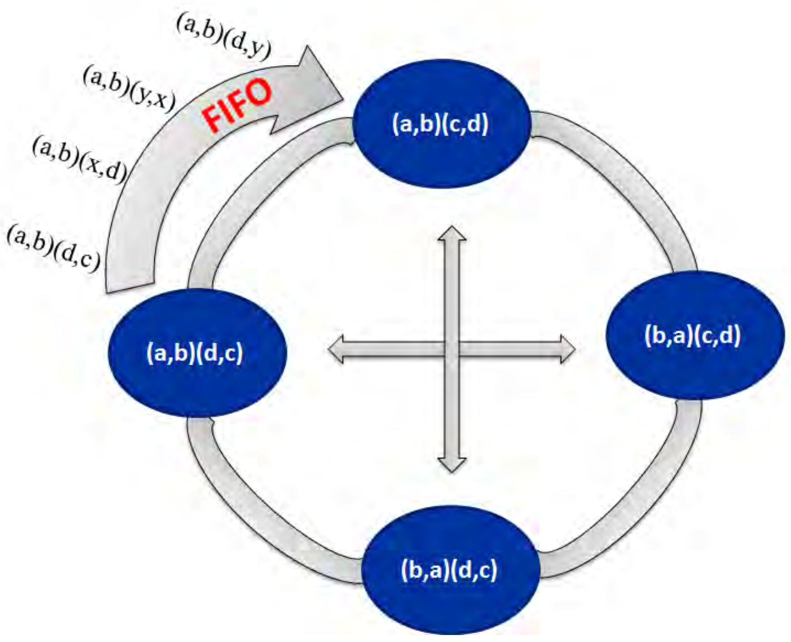


Figure 2.17. LFU/LRU state transition [109]

Moreover, cache management for ICN using LNC has been considered to improve the network throughput (i.e., to reduce the cost of network bandwidth by jointly considering content routing and caching strategy). On the other hand, reducing the utilization of the network bandwidth, network security is a major challenge for communication networks. Likewise, the unreliability of the network link leads to content packet loss that results in the decoding failure.

2.3.15 Time Aware Least Recently Used (TLRU)

LRU is a widely deployed cache management policy. It maintains contents in cache according to their access time. If there is no free cache space and a new content arrives, it deletes the least recently used content to make space for the new arrived one. On the other hand, LFU policy sorts the queue in terms of access frequency. If the cache space is full and a new content arrives, it replaces the lowest frequency content of the queue.

However, due to the nature of LRU and LFU, there is a possibility of evicting the most popular content by a least popular one. To overcome this problem, Time Aware Least Recent Used (TLRU) policy was proposed in [109]. It is a popularity-based content life time aware eviction policy, which is an extension of the simple LRU. In this policy, the time stamp of an arriving content is calculated locally by a cache node. The arriving content is cached if the average request time is smaller than the time stamps of the stored contents. TLRU stores the content if space is available in cache otherwise it applies simple LRU on the cached contents to create space for the new arriving content. Unlike LFU and LRU which are non-protective eviction policies, TLRU is proposed to evict relatively less popular contents. In this policy, as shown in Figure 2.17, the eviction order of the cache state can be changed at any instant of time without changing the stored contents in cache. In the given figure, the arrow represents one state transition of First-In-First-Out (FIFO) for some contents.

The authors claim that TLRU outperforms LRU and LFU if space is available in cache otherwise the simple LRU policy is used to create space for the new arriving content. But according to Cisco Visual Networking Index [11], as the delivery of multimedia applications is becoming extremely popular on the Internet, the aggregation of all forms of video will be roughly 86 percent of the worldwide user traffic. Due to this exponential increase in the number of Internet users and demand for multimedia applications, there is a vivid chance of filling cache of CRs in a network very rapidly. In that case the TLRU will be no longer effective for content eviction.

2.3.16 WAVE

In WAVE [110], chunks are adjusted in cache based on the popularity value of the contents. The basic idea of WAVE is that, as illustrated in Figure 2.18, the chunks are organized based on the access count of the contents. Upon increasing the access count, the number of chunks to be stored is increased exponentially and disseminated more

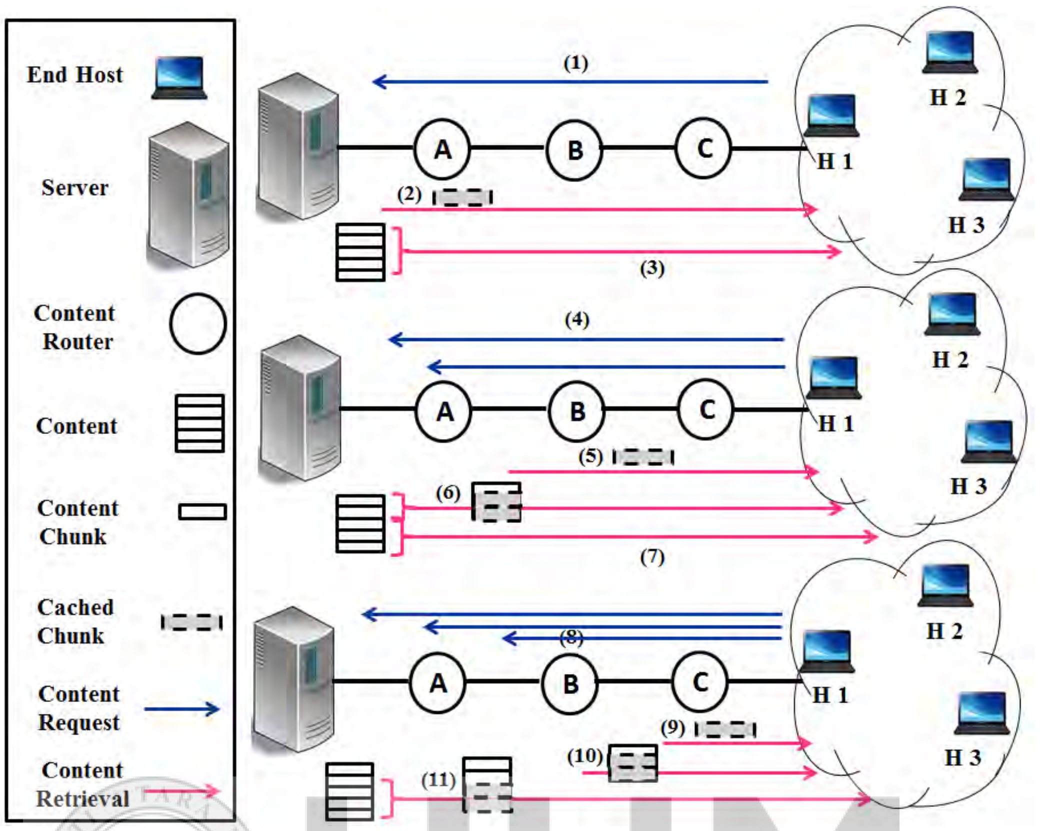


Figure 2.18. Illustration of WAVE Operations [110]

widely. In addition, WAVE does not require any prior knowledge of access patterns, therefore, it can be operated with any content routing strategy (as it follows merely the information from where it receives the content request). As no central server is used in this scheme, caching decisions are made independently at each individual router. Moreover, the number of content items is recommended to a downstream router by its upstream one, which is increased exponentially as the number of requests increases. If the downstream router is unable to cache the content by any reason, it simply ignores the suggestion and recommends the content to other routers on the same path.

In WAVE, an upstream router suggests what content chunks to be stored in its next downstream router. Therefore, as content is becoming popular, replicas of the corresponding chunks are becoming closer to the end users [93]. Although WAVE is an efficient caching scheme, it cannot eliminate the caching redundancy and is difficult to achieve good caching performance locally [34]. In other words, as distance grows, the performance is dramatically decreased [111] because content servers are involved in the recommendation [112]. Furthermore, for the adjustment of the number of contents to be cached, WAVE takes benefit of the popularity property of the content at each site in ICN and disseminates the contents in a hop-by-hop fashion. While with the exponential growth in mobile networks, the introduction of mobile ICN is of great interest in terms of content properties, for example, popularity, geographical and social characteristics. But these features (i.e., content characteristics and wireless access of mobile users) are ignored in WAVE [113].

2.3.17 Most Popular Content (MPC)

MPC [33] is developed for the CCN caching, where routers store only popular contents. In this strategy, every router locally calculates the number of requests for each content name, and caches both content name and popularity count into a *Popularity Table* (PT). The content name is labeled as being popular if it reaches a *Popularity Threshold* locally. And if a router holds the content, it suggests its neighbors for caching it with a new *Suggestion* message (see Figure 2.19). In the given figure, three subscribers - X, Y, Z, access content *d* and one subscriber - X, accesses content *c*. In the PT, the popularity value of content *d* is 3 whereas the popularity value of content *c* is 1. It means that content *d* is popular and is therefore suggested to neighbor routers B and C for caching. The suggestion may or may not be accepted due to local policies, for instance, resource availability. In order to prevent flooding (for example, if the content popularity decreases with time after the suggestion process), MPC reinitializes the popularity count according to a *Reset* value.

With storing only popular contents, MPC reduces searching overhead during caching operations and in turn achieves a higher hit rate in comparison to LFU and LRU. De-

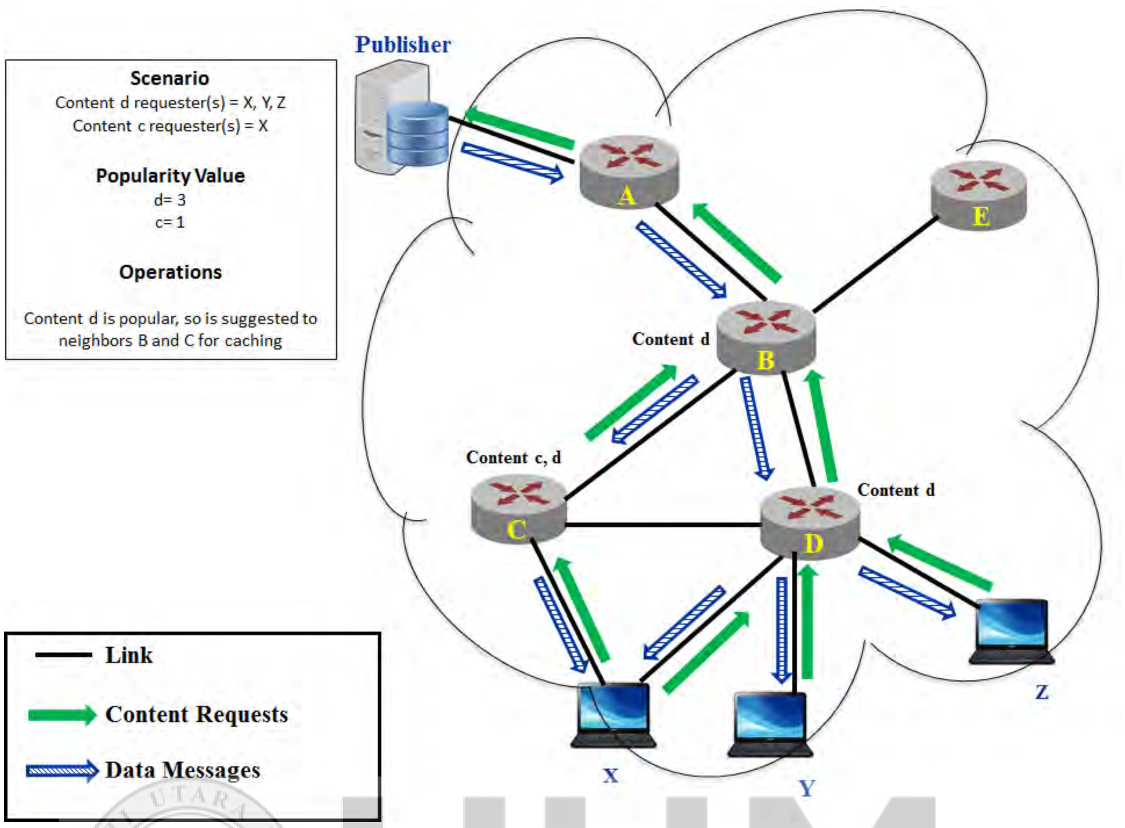

Scenario
Content d requester(s) = X, Y, Z
Content c requester(s) = X
Popularity Value
d= 3
c= 1
Operations
Content d is popular, so is suggested to neighbors B and C for caching
Publisher
A
E
B
Content d
Content c, d
C
Content d
D
X
Y
Z
Link
Content Requests
Data Messages

Figure 2.19. MPC Workflow Example

spite the fact that MPC achieves some higher cache utilization, the bandwidth and memory consumption during *Suggestion* messages is typically high. Also, it has no policy for content eviction, so the content may simply be deleted (regardless of its popularity) upon arrival of the new one. Consequently, subsequent requests for the deleted content will be forwarded to the server and thus will affect the network efficiency with respect to bandwidth utilization.

2.3.18 Fine-Grained Popularity-based Caching (FGPC)

Fine-Grained Popularity-based Caching (FGPC) [114] strategy caches all incoming contents if content store (CS) of network node is not full. Otherwise, it stores only popular contents. Based on FGPC, a Dynamic-FGPC (D-FGPC) [114] was proposed which regularly modifies the content popularity threshold. When forwarding a content to downstream or receiving from upstream, three kinds of statistic information are

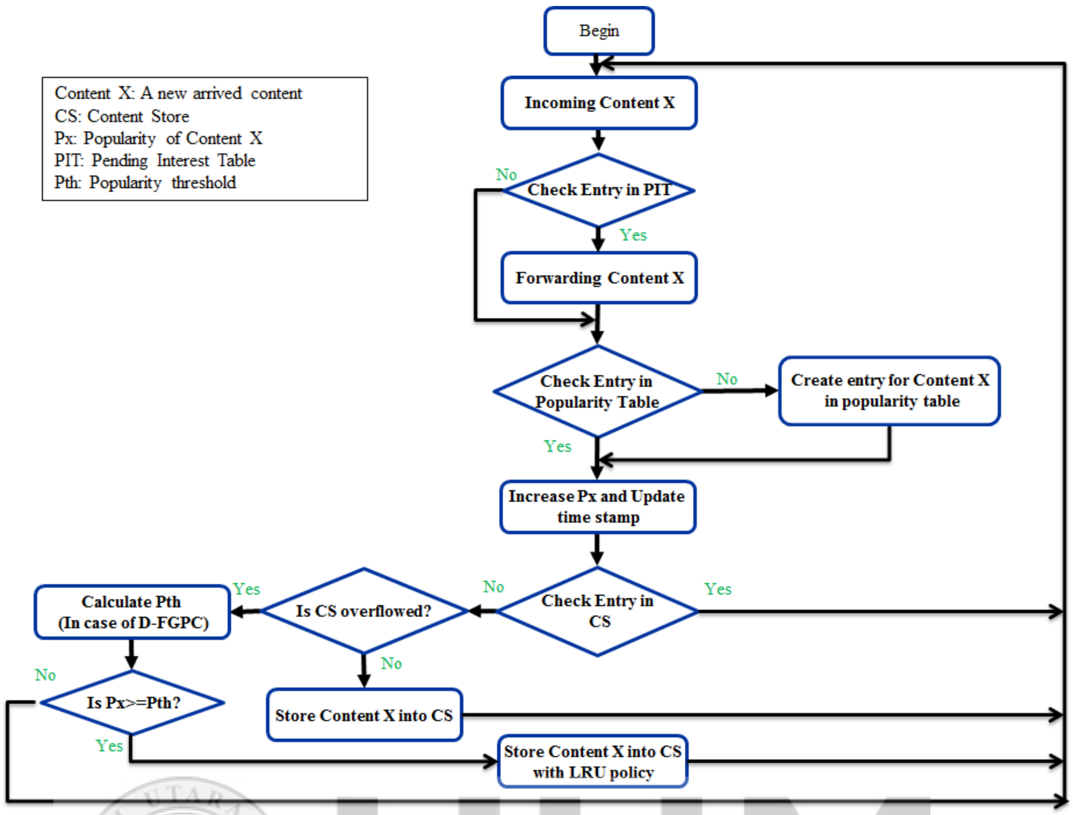

Content X: A new arrived content
CS: Content Store
Px: Popularity of Content X
PIT: Pending Interest Table
Pth: Popularity threshold
Begin
Incoming Content X
Check Entry in PIT
No
Yes
Forwarding Content X
Check Entry in Popularity Table
No
Create entry for Content X in popularity table
Yes
Increase Px and Update time stamp
Check Entry in CS
Yes
No
Is CS overflowed?
Yes
No
Calculate Pth (In case of D-FGPC)
Is Px>=Pth?
No
Yes
Store Content X into CS
Store Content X into CS with LRU policy

Figure 2.20. FGPC and D-FGPC Flowchart [114]

updated in this strategy, i.e., content counter, content name, and time stamp. When a content store becomes full and a new content arrives, its popularity is compared with the popularity threshold value (see Figure 2.20). If the value of new content is greater than the predefined threshold value, FGPC adopts the LRU policy to cache new content into content store, otherwise ignores it.

Although FGPC outperforms some of the existing strategies but as it caches all incoming contents, searching overhead is increased during the replacement process. This is just because if a content is not popular then it does not contribute in the hit rate, therefore, storing it in the memory increases the computational overhead for the replacement policy. Moreover, D-FGPC changes the threshold value according to the popularity of the content, i.e., if the popularity of content increases, D-FGPC also increases the threshold value. For instance, if the popularity of content reaches 1,000,

Table 2.1
Analysis of Caching in ICN Architectures

Architecture	Reference	Off-Path Caching	On-Path Caching
DONA	Koponen et al. [18]	Requires additional registrations.	Through resolution handlers.
NDN	NDN project [21]	Requires additional routing information.	At content routers.
MobilityFirst	MobilityFirst project [20]	Requires additional registrations.	At content routers.
CONVERGENCE	CONVERGENCE project [26]	Through unspecified NRS.	At content routers.
COMET	COMET project [28]	Requires additional registrations.	At content routers.
SAIL	SAIL project [27]	Requires additional routing information.	At content routers.
PURSUIT	PURSUIT project [25]	Requires additional registrations.	Through resolution handlers.

the threshold value will also be set as 1,000. Consequently, all those contents which have less than 1,000 popularity will never be cached, and thus the overall cache hit rate of contents is reduced.

Besides the above mentioned caching strategies, two analytical caching models have been proposed using Markov modulated rate process in [115] and Markov chains in [116] to evaluate data transfer in CCN. Moreover, an analytical model for storage and bandwidth sharing is proposed in [117] to guide a trade-off between the limited network resources and user performance. Additionally, some experimental assessments of cache management in ICN are presented in [118]. Some other research on caching has been proposed in [119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131].

Table 2.2
Summary of Characteristics of the ICN Off-Path Caching Strategies

Caching Strategy	**Contributions**	**Limitations**
Hash-routing [96]	Reducing caching redundancy, as the hash function indicates that the content is cached wherever it is found.	Increasing the possibility of intradomain link load in both multicast hash-routing and symmetric approaches.
CIC [97]	Improving the caching diversity by storing different kinds of chunks and boosting the average response time because of the availability of the chunks near to the customers.	Increasing computational overhead with respect to searching time if some chunks need to be deleted for creating space for a new arrived content.

All the above mentioned works however do not investigate any optimum cache management strategies. In reality, placing the contents dynamically in appropriate intermediate nodes is a significant and challenging task. The caching analysis in different ICN architectures is presented in Table 2.1 and a summary of the presented off-path and on-path caching strategies are expressed in Table 2.2 and Table 2.3, respectively.

2.4 Summarization of Caching Strategies

ICN caching consists of two approaches: off-path caching and on-path caching. Off-path caching is analogous to the CDN server placement or traditional proxy caching. In this approach, the old items are replaced with the new arrived ones after contacting the NRS, which increases the network overhead. On the other hand, on-path caching is opportunistic, where network nodes cache the content if they receive its copy. Consequently, if a new request arrives for the same content, the network node satisfies it locally without contacting the NRS. Caching copies of the contents locally may reduce the communication and computational overhead. However, it is challenging to decide that which content should be cached and what location would be suitable for caching so that the ICN performance can be enhanced in terms of cache hit rate, content redundancy and retrieval delay. For this reason, several strategies have been proposed in the literature.

As stated earlier that off-path caching is similar to the current proxy system, it received less attention from the research community, but a few schemes have been proposed such as Hash-routing schemes and CIC. In Hash-routing schemes the edge node calculates the hash function upon receiving a request. If the hash function matches the content identifier of a particular router, the request is forwarded there. If the router has already cached the requested content, it replies with a copy of it, otherwise the request is forwarded to the main server. CIC is another off-path caching strategy, where a content is divided into several chunks and those chunks are placed at more than one location in the network.

On-path caching is a popular approach that has attracted the majority of the research community and therefore various strategies have been designed and proposed for this particular type of caching. The first on-path caching strategy was LCE, which is also known as the default ICN caching strategy. In LCE, a copy of the content is stored at all nodes on the publisher-subscriber path. Prob caching strategy is a modified version of LCE, but the main difference is such that Prob caches the object copy with probability. LCD caches a copy of the requested object at the node located immediately below the node where hit occurs. MCD is similar to LCD in functionality, but if hit occurs at a node then the copy is deleted from that particular node and is shifted to the underlying cache. ProbCache includes TSI and TSB fields which calculate probability of the contents. The main goal of ProbCache is to place the contents near the subscriber.

In Breadcrumbs after downloading a content, a trail is created and maintained at each router for caching the routing history. In CLS, which is the modified form of Breadcrumbs, a trail is also created for maintaining the routing history. However, in CLS the trail is created at the time of content caching unlike Breadcrumbs which creates the trail after the content is forwarded to the underlying router. OCPCP caches the contents based on their popularity value which is calculated during content caching. In CATT strategy, on the edges of an AS some CATT nodes are deployed as caching nodes and the contents are cached at a single location on the publisher-subscriber path. Betweenness-Centrality caches contents at a node which has the maximum number of routes traversing it. One-touch caching scheme stores the contents at the server based on their access time. In case of memory overflow, the contents are randomly replaced with the new arrived ones.

NCCM scheme is based on SDN and jointly considers content routing and caching through LNC. TLRU is a popularity-based content eviction policy where a local node calculates time stamps of the arrived contents. If the average request time of the arrived content is smaller than the time stamp of the cached contents then it is recommended for caching. WAVE is a popularity-based caching strategy which organizes the chunks on the basis of their access time. MPC also caches contents based on their popularity value, however, it introduces a popularity table for storing content name and its popularity value. In MPC, the neighbor nodes are suggested for caching a content when its popularity value reaches the popularity threshold. FGPC is the modification of MPC that also includes a popularity table for caching both the content name and its popularity value. However, the threshold in FGPC is regularly modified based on the content popularity value.

2.5 Summary

The detailed background of the issues that are covered in this research was provided in this chapter. The background materials on cache management strategies, i.e., off-path caching: Hash-routing schemes and CIC, and on-path caching: LCE, Prob, LCD, MCD, ProbCache, Breadcrumbs, CLS, OCPCP, CATT, Betweenness-Centrality, One-touch Caching, NCCM, TLRU, WAVE, MPC, and FGPC, were reviewed and critically analyzed for significance. Moreover, this chapter presented the ICN architectures for content caching, which produce foundation for the theoretical framework of this research. Additionally, it revealed the necessity for flexible caching strategy that can effectively place and evict content in ICN. In the next chapter, the research methodology for achieving the objectives of this research and evaluating the FlexPop strategy will be explored.

Table 2.3
Summary of Characteristics of the ICN On-Path Caching Strategies

Caching Strategy	Contributions	Limitations
LCE [32]	Increasing hit ratio by storing a copy of the requested object in all intermediate nodes on the path from the node on which hit occurs through the requesting user. Caching content near to the user and thus reducing content retrieval delay and stretch.	Increasing redundancy by making the cache of all intermediate nodes full.
Prob [32]	Reducing content redundancy by storing a copy of the requested object on the path from the location of the hit to the requesting host.	Minimizing hit ratio if the cache of all intermediate nodes become full and no space is available for the new arriving contents.
LCD [32]	Reducing content redundancy by storing a copy of the requested object at the node which is located immediately below the node where hit occurs.	Overwhelming the cache of nodes on the link in case the popular contents are requested frequently. Utilizing maximum bandwidth if the new interests do not find the requested content(s).
MCD [100]	Reducing the number of copies for the same object between the server and the requesting host by shifting the requested object to the underlying cache if the hit occurs at level *l*.	Increasing content request delay because if a content is evicted from the cache and it is requested again then it can only be replied from the server.
ProbCache [41]	Maximizing the number of different content items stored along a delivery path to reduce redundancy between network caches.	Decreasing *hop decrement* because it frequently replaces contents at nodes near to the user(s).

Caching Strategy	Contributions	Limitations
Breadcrumbs [101]	Maintaining routing history by creating a trail at each router that helps to reduce the server overhead.	Maximizing content delay during downloading because if the file is evicted at each router, the request may suffer missing downstream.
CLS [102]	Maintaining routing history by creating a trail at each router, which guarantees that the CLS saves cache space by deleting a trail on time, and hence, finds the chunk when search downstream.	Maintaining redundancy and increasing content delay (as it cannot ensure the placement of chunk at the nearest node to the user).
OCPCP [103]	Minimizing content redundancy and retrieval delay and improving the network throughput in terms of bandwidth utilization as the requests can be satisfied locally instead of forwarding to the server (publisher).	Increasing the chances of filling router's cache in a network very rapidly due to exponential growth in the number of Internet users, and hence reducing the cache hit ratio as excessive content replacement may happen.
CATT [104]	Minimizing content retrieval delay and redundancy by caching contents only at the edge routers of autonomous systems (ASs).	Reducing cache hit ratio if the cache of edge routers overflow and the new contents need to replace the cached ones.
Betweenness-Centrality [106]	Minimizing redundancy by storing content objects only once along the path, particularly at the router having the maximum number of shortest routes traversing it.	Lacking mobility, i.e., difficult for a router to find the value of its betweenness centrality if it is deployed on mobile ad-hoc networks.

Caching Strategy	Contributions	Limitations
One-touch caching [107]	Reducing computational complexity by preserving the most recently requested contents in the server, and evicting the existing contents randomly if there is no free cache space.	Utilizing maximum bandwidth during request to and/or reply from the servers (as all requests are served by content servers).
NCCM [108]	Minimizing the network bandwidth cost by reporting content request statistics periodically with a fixed time period by all CRs.	Causing decoding failure because the unreliability of the network link leads to content packet loss.
TLRU [109]	Caching arriving content if the average request time is smaller than the time stamps of the stored contents and therefore increasing cache hit rate.	Boosting the chance of filling cache of content routers in a network very rapidly and hence reducing its effectiveness for content eviction.
WAVE [110]	Increasing the cache hit ratio because of content popularity.	Decreasing the performance as it cannot eliminate caching redundancy.
MPC [33]	Storing popular contents which helps to improve cache hit ratio; reducing network resource utilization; and decreasing content redundancy.	Consuming maximum bandwidth and memory during suggestion messages.
FGPC [114]	Achieving higher hit rate and fast convergence speed till the moment the content store has the capacity to store incoming contents.	Reducing the cache hit rate if the content store is full, as the threshold value is dynamically modified according to the popularity value of the most popular content.

CHAPTER THREE
RESEARCH METHODOLOGY

The goal of this research is to design and evaluate a flexible popularity-based caching strategy for ICN. The core objectives to be addressed in this chapter are validation, verification, and performance evaluation of the FlexPop strategy. To accomplish this task, it requires a particular methodology to follow, which is the center point of this chapter. The research adopts the methods reported in the Design Research Methodology (DRM) by clearly describing each step of the approach. To present the aim of this chapter, a number of subsections have been prepared. It starts by presenting the available research methodology options and research steps as described in Section 3.1. The first stage of DRM, known as Research Clarification (RC), is presented in Section 3.2. This section presents the goals of RC stage with respect to the adopted methods and deliverables. Descriptive Study-I (DS-I), which is the second stage in DRM, is elaborated in Section 3.3. The DS-I stage describes the steps to acquire adequate comprehension regarding the present situation and actualizes a conceptual model. The methods adopted in designing the proposed flexible caching mechanisms are discussed in the third stage of DRM, known as Prescriptive Study (PS) which is presented in Section 3.4. Evaluation techniques and performance metrics are associated with the fourth stage of DRM, called Descriptive Study-II (DS-II), which is discussed in Section 3.5. The summary of all methods and approaches discussed in the chapter is presented in Section 3.6.

3.1 Research Approach

The main purpose of this research is to design a strategy for content placement and eviction in ICN, with a goal to enhance the overall performance of caching in terms of increased cache hit rate, reduced redundancy, content retrieval delay, stretch ratio, and efficient utilization of memory. Although, this is a challenging issue to map the existing caching schemes to the new one leading to an efficient and optimum solution, these requirements are fitted with the design research definition as proposed by Blessing and Chakrabarti [132]. These features complement each other in order to produce an efficient and effective solution, which would result in a better performance [133].

According to Blessing and Chakrabarti [132], design research must be scientific in obtaining appropriate results both in theoretical and practical sense, therefore, it needs a particular method for its exclusive characteristics. For this purpose they have proposed a method called Design Research Methodology (DRM). In addition, it is submitted in [134] that a scientific methodology for computer science should consist of modeling, conceptualization, experiments, and simulation. The DRM follows the same concept and helps making design research more precise and economical, therefore, it has been adopted for conducting this research.

DRM can be categorized in the following four stages:

- Research Clarification (RC)
- Descriptive Study-I (DS-I)
- Prescriptive Study (PS)
- Descriptive Study-II (DSII)

A brief explanation of DRM stages from the perspective of this research area is presented in the following sections. Figure 3.1 demonstrates the DRM framework where it shows the links between DRM stages, the techniques adopted in each stage, and the major deliverables [133]. The main flow of process is shown by light arrows between the stages, while the bold arrows show adopted techniques and deliverables of each

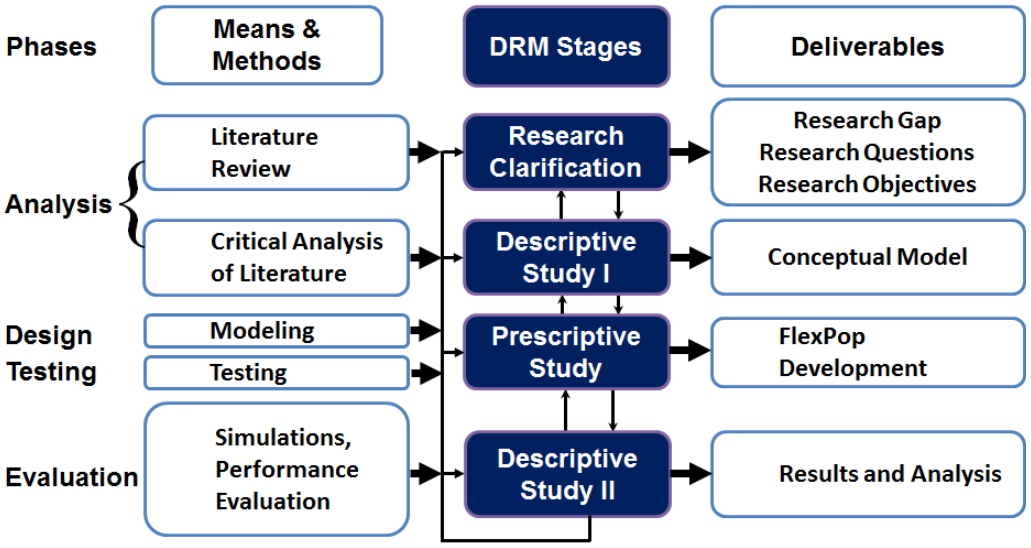


Figure 3.1. Research Approach [132]

individual stage.

3.2 Research Clarification (RC)

Research Clarification (RC) is the first stage of DRM which is used to get a specific knowledge about the overall research plan. The RC includes six iterative steps, as shown in Figure 3.2.

The deliverable of this research in the RC stage is the overall research plan which consists of the following points [133]:

- The basis of research and its motivation
- Research questions, objectives, and problems
- Important areas to be addressed
- Research approach
- Areas of contributions.

In this study, the main deliverables of RC stage were such that the literature was thor-

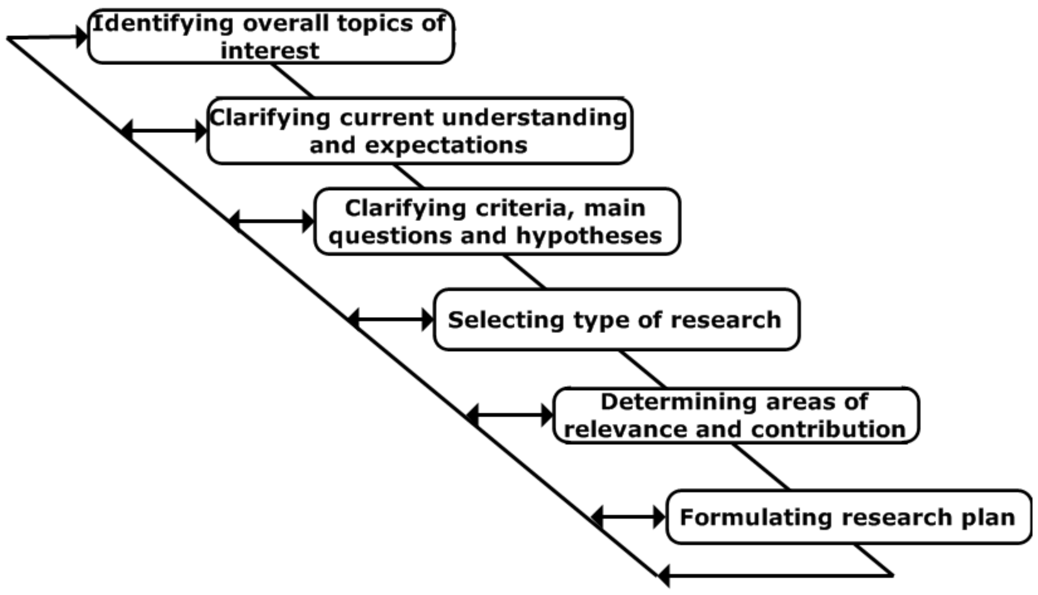

Identifying overall topics of interest
Clarifying current understanding and expectations
Clarifying criteria, main questions and hypotheses
Selecting type of research
Determining areas of relevance and contribution
Formulating research plan

Figure 3.2. Main Steps in the Research Clarification Stage [132]

oughly reviewed and on the basis of that the research gap was found. Moreover, motivation of ICN as the future Internet architecture was elaborated, the problem statement was defined, and research questions and objectives were clearly described.

3.3 Descriptive Study-I (DS-I)

The second stage of DRM is DS-I which is used to achieve a deep knowledge of the current situation. This stage includes critical review of the existing work as well as empirical studies in the research field. During the overall duration of this research, a comprehensive review of the existing cache management strategies was summarized. Many empirical studies were also critically evaluated to achieve deep knowledge of the existing cache placement and eviction strategies. The DS-I stage includes five steps, i.e., literature review, research scope identification, research plan, empirical study, and overall conclusion [133], as illustrated in Figure 3.3.

The deliverables of DS-I stage for this study include:

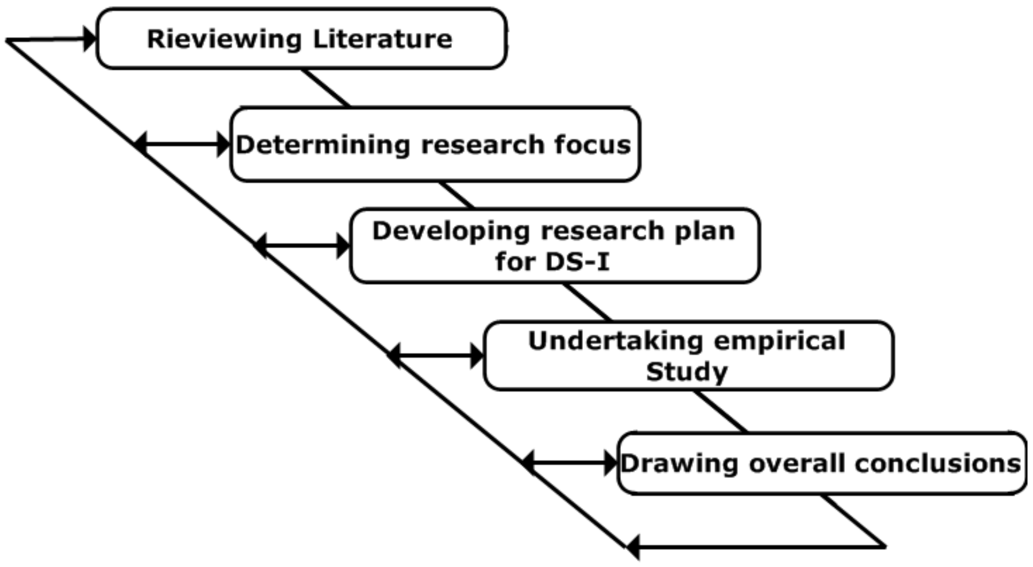

Rieviewing Literature
Determining research focus
Developing research plan for DS-I
Undertaking empirical Study
Drawing overall conclusions

Figure 3.3. Main Steps in the Descriptive Study-I Stage [132]

- Performing a comprehensive study on cache management strategies in ICN, for example, content placement, content eviction, and suitable node for caching.
- Critical review of the existing cache management strategies in terms of their contributions and limitations, which lead to the conceptual model design of FlexPop strategy.
- Performing an inclusive analysis on the algorithms in FlexPop and executing simulation measurement to recognize the strengths and weaknesses of the algorithms. Also, to identify the research challenges in ICN caching that should be taken into consideration during analyzing caching strategies.

3.3.1 Conceptual Model of FlexPop

In order to improve cache performance in ICN while basing on the proposed performance model and the critical review of existing strategies, *cache hit rate*, *path redundancy*, *retrieval delay*, *memory utilization* and *stretch* are the major points to be addressed. As an outcome of addressing ICN caching, a conceptual model of FlexPop is designed to describe the expected desires and improved situation. The FlexPop architecture and proposed conceptual model for research objectives are illustrated in Figure

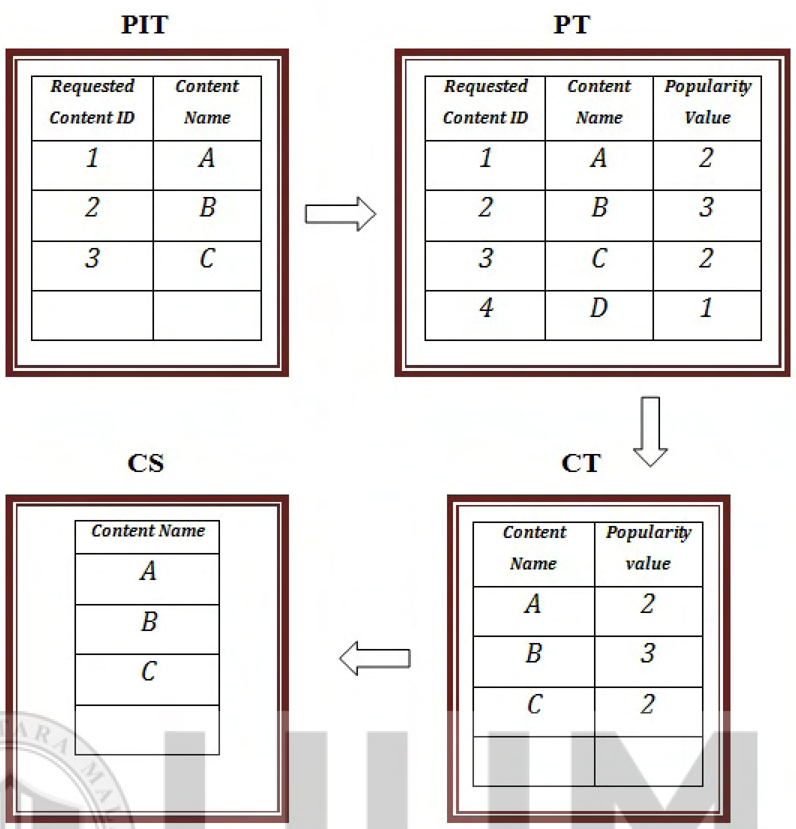


Figure 3.4. FlexPop Architecture

3.4 and Figure 3.5, respectively. In the given figures, when a new request arrives, its entry is checked in the Pending Interest Table (PIT) - the default ICN data structure. If the entry is found in the PIT, it is forwarded to the Popularity Table (PT). PT calculates the number of request popularity based on their access count. If the request popularity reaches the threshold value, i.e., $\geqq 2$, it is forwarded to the Comparison Table (CT). Finally, the content is cached at a router which has the highest number of outgoing interfaces in the network. The full explanation of conceptual model is given in full detail in Chapter Four.

Our proposed system is based on Multiple Attribute Decision Making (MADM) approach [135], which is explained in the following subsection.

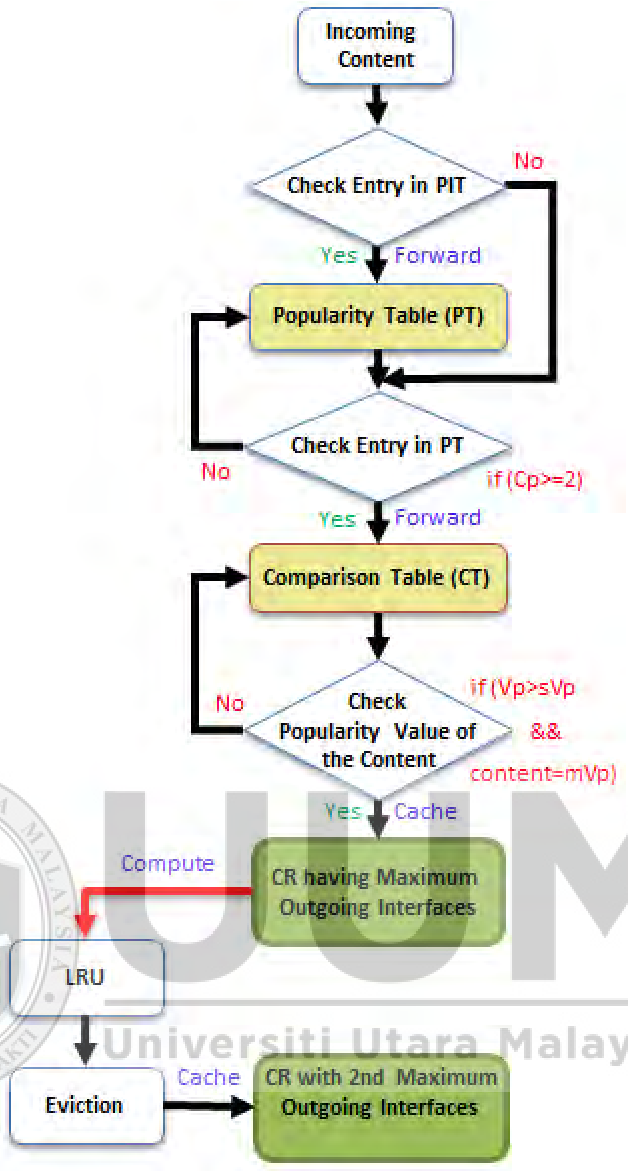


Figure 3.5. Conceptual Model of FlexPop

3.3.2 Multiple Attribute Decision Making (MADM)

In the literature [136, 137], MADM is one of the effectively utilized techniques for solving decision making problems. MADM is related to the the problems whose number of alternatives has been predetermined [137]. Several MADM techniques are available for attributes selection in making a decision such as Conjunctive technique, Disjunctive Method, Linear Assignment Method (LAM), SAW, TOPSIS, and so on. According to [135], in Conjunctive technique the alternative is rejected which does not qualify the minimum acceptable level for all attributes. In Disjunctive technique an alternative is selected which has the highest value among all attributes. In LAM, that alternative which has several high value attributes is considered as high ranked alternative. In SAW method, the total values of an alternative is calculated and the one with the highest attribute values is selected. In TOPSIS, the selected alternative must have the longest distance from the negative-ideal solution and the shortest distance from the positive ideal solution. In our proposed strategy the contents are selected based on their popularity values, therefore, according to the above definitions, SAW is the suitable method for the first part of FlexPop, called CPM. Similarity, upon the arrival of the new contents, when the memory is full then one of the cached contents is evicted and placed at another location which will be nearer to the users. In this situation TOPSIS is the most suitable technique to be followed. Therefore the second part of FlexPop, named CEM, is based on TOPSIS method. The adopted SAW and TOPSIS methods are presented in Chapter Four with detail explanation.

3.4 Prescriptive Study (PS)

Prescriptive Study (PS) is considered the essential phase during conducting a research as it deals with the development of the proposed mechanisms [133]. For the benefit of this study, for more comprehension, modeling and simulation, the proposed process of Guizani et al. [138] was followed. The illustration of PS stage regarding the FlexPop caching strategy is demonstrated through Figure 3.6.

The first block illustrates the specifications of CPM and CEM mechanisms to be modeled. The second block is related to the conceptual/analytical modeling which includes problem formulation, objectives determination, and literature review. Furthermore, it generally consists of suppositions and presenting an overview to reduce complexity of the proposed model. The third block deals with the design of proposed algorithms, while the fourth block shows model implementation which depends on the choice of

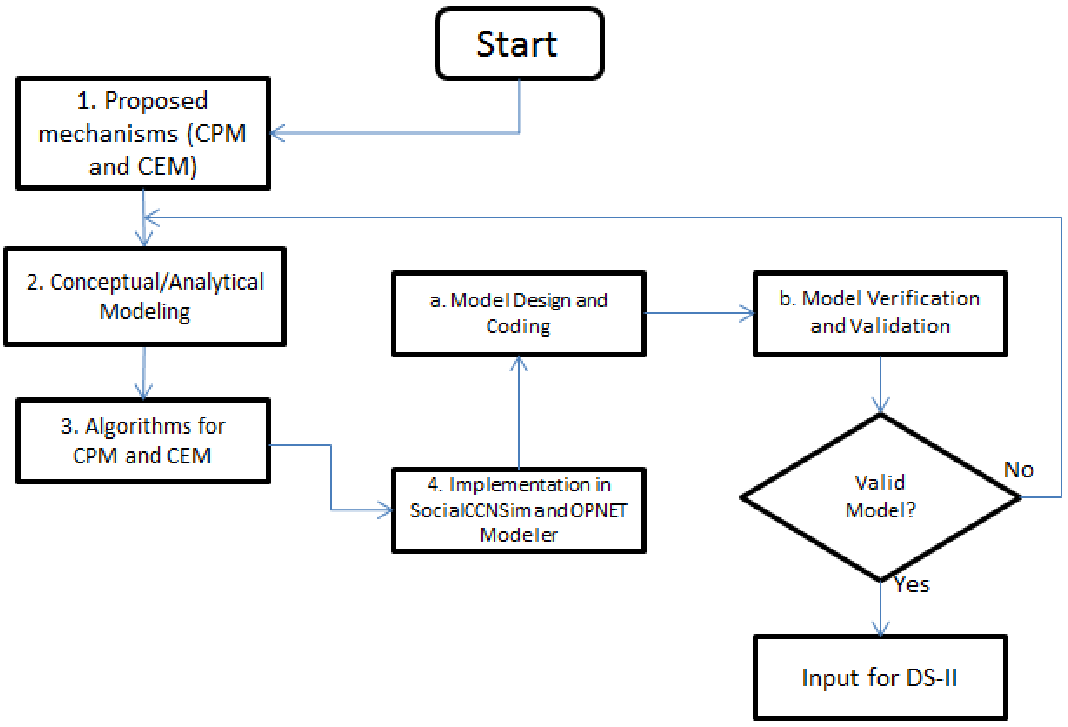

Start
1. Proposed mechanisms (CPM and CEM)
2. Conceptual/Analytical Modeling
3. Algorithms for CPM and CEM
4. Implementation in SocialCCNSim and OPNET Modeler
a. Model Design and Coding
b. Model Verification and Validation
Valid Model?
No
Yes
Input for DS-II

Figure 3.6. Prescriptive Study Steps [138]

the simulation environment. Finally, validation and verification of the proposed strategy are covered in more detail in the following subsection.

The deliverables of the PS stage are:

- Development of the proposed strategy (discussed in Chapter Four and Five),
- Design and implementation of the proposed mechanisms, i.e., CPM and CEM, and
- Validation and verification of FlexPop strategy.

3.4.1 Validation and Verification

Validation and verification are defined as a process of verifying the accuracy and internal consistency of data, and confirming that it embodies the real-world entities suitable for its expected goal or intended number of objectives [139]. According to the modeling and simulation community [140], validation is the procedure to find out the level

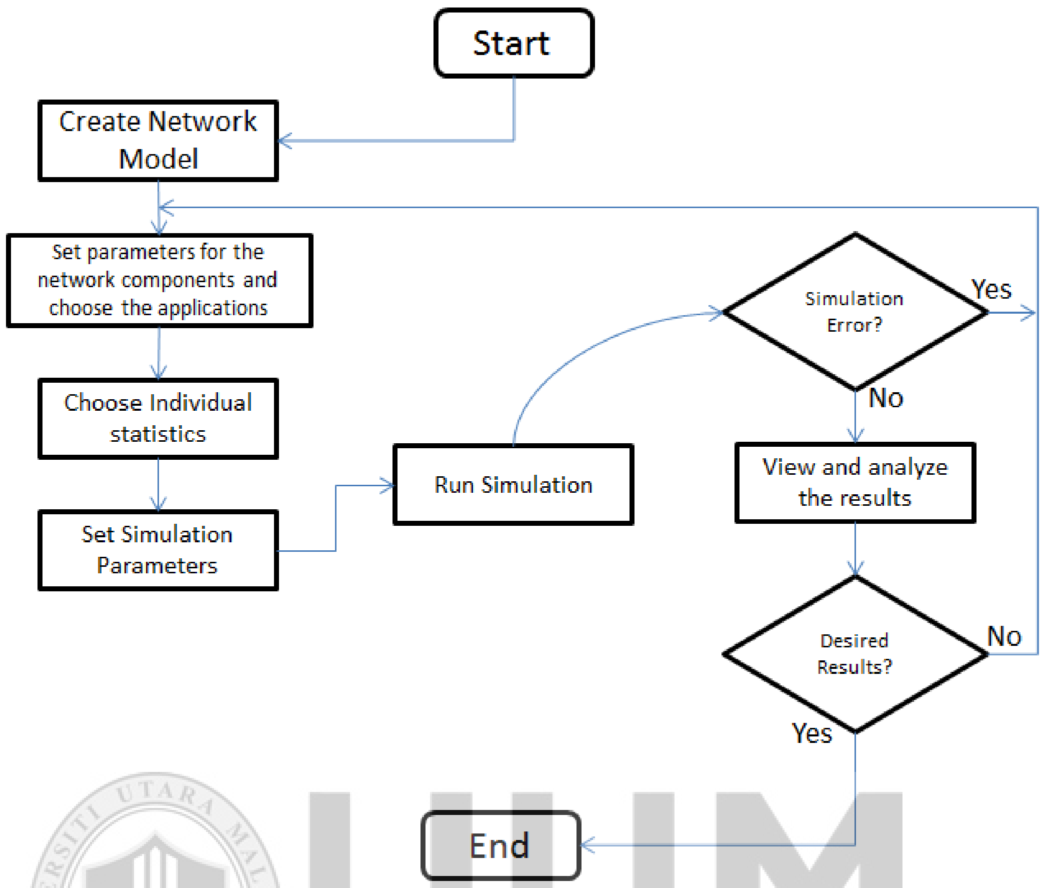

Start
Create Network Model
Set parameters for the network components and choose the applications
Choose Individual statistics
Set Simulation Parameters
Run Simulation
Simulation Error?
Yes
No
View and analyze the results
Desired Results?
No
Yes
End

Figure 3.7. Main Steps for Validation and Verification [141]

to which a prototype, simulation, or combination of prototype and simulation and their allied data precisely represent the real world from the perspective of its expected use(s). While verification is the procedure to find out that a prototype, simulation, or combination of prototype and simulation and their allied data precisely represent the developer's conceptual model and its explanation. Thus, it is a step for ensuring the accurate assumptions of the model implementations. This includes evaluation of the simulation program to guarantee that the conceptual model is a true reflection of the operational model [133].

Figure 3.7 shows the main steps for validation and verification of this research. That is, the network model was created in the two simulators - SocialCCNSim [142] and OPNET Modeler 18.0 [143]. Then the following parameters: cache size, catalog size, Zipf probability model - α, and number of simulations, were set and the individual statistics were chosen. For the purpose to validate the simulator we tested the four ICN caching strategies - LCE, LCD, ProbCache, and Betweenness-Centrality with respect to cache hit rate. Detailed simulation results and discussion are presented in the following section.

3.5 Descriptive Study-II (DS-II)

This stage focuses on the evaluation of the proposed strategy. Performance evaluation is the imperative phase in evaluating any research [133]. The following research steps were carried out to accomplish this goal:

- Executing the proposed system using SocialCCNSim and OPNET Modeler 18.0 network simulators.
- Comparing performance of the proposed strategy with those of LCE, ProbCache, and MPC.

3.5.1 Evaluation of Communication Network System

Generally, to evaluate the performance of communication network system, three different approaches are used, i.e., mathematical/analytical modeling, measurement, and simulation [144][145]. Figure 3.8 demonstrates the above three methods for evaluating the performance of communication network.

The mathematical/analytical modeling method is a set of equations that represents the procedure of a communication network system under study using mathematical symbols. The method includes designing, determining, and validating the analytical model of the system to investigate and explain the problem [146]. This may lead to an improved declaration of the system before the real implementation starts.

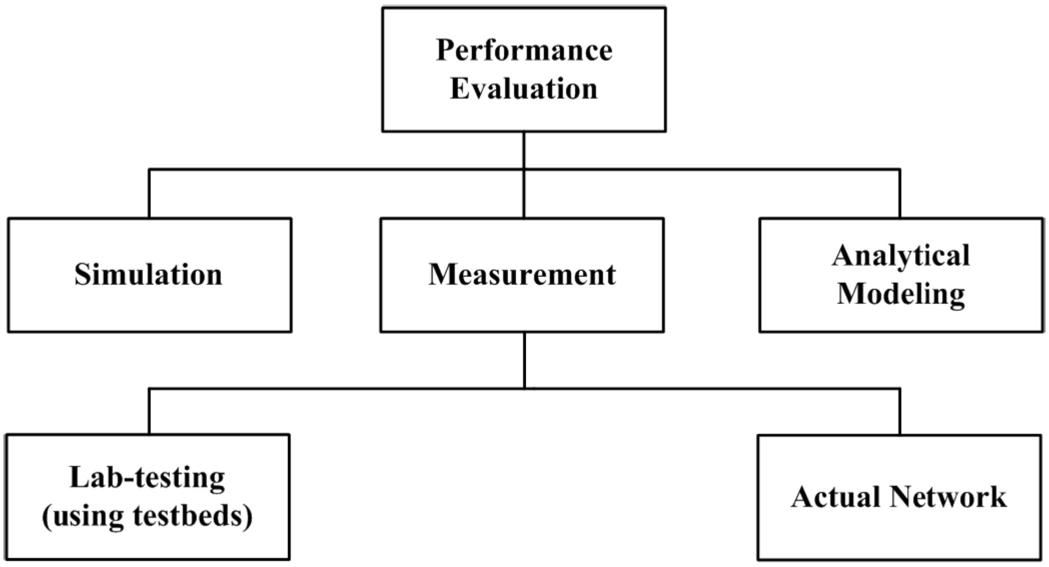


Figure 3.8. Performance Evaluation Approaches [145]

The second method for evaluating network performance computes prototype of the communication system [147]. Using this approach, prototypes of a communication network system can be designed and tested in a particular network environment. Measurement of network traffic provides with a way of understanding that what is or is not working properly in a local-area or wide-area network [148].

The third method for evaluating network performance is simulation, which is an abstraction or estimation of the real world [140]. Simulation is an experimentation of an object that includes devices, tools, systems, or subsystems in a realistic form [149].

In this way, the researchers are allowed to carry out analyses on the procedures of the targeted network model without disturbing the actual network. In addition, simulation is an exercise of developing and analyzing a scheme of theoretical communication network system using network simulation software [146].

3.5.2 Network Simulators

The most popular available network simulators are ns-2 [150] and ns-3 [151]. These simulators are based on two languages, i.e., OTcl and C++. OTcl is actually an object oriented version of Tcl. OTcl is used for configuration and simulating scripts and C++ is used to model the behavior of the protocols. The implementation of a protocol in ns-2 and/or ns-3 is such that first binding is created between the OTcl and C++, and then in the OTcl script the scenario of the simulation is described. After that, the simulation is simply run and the generated trace files are analyzed. Because of the OTcl scripts, ns-2 and ns-3 are somehow difficult to understand. Also, due to the combination of OTcl scripts and C++, debugging is complicated in ns-2 and ns-3 [152].

Likewise, SocialCCNSim and OPNET Modeler are other useful network simulators; all the scenarios in this work were tested in these two simulators, therefore, their features are discussed in the following subsections.

3.5.3 Simulators for ICN

Different ICN projects have developed different simulators for testing the performance of their respective architectures. The most commonly used simulators are CCNx [153, 154], ccnSim [155], ndnSim [156], and ICN Simulator [157]. CCNx is an emulation testbed, which is suitable for some of the operations (e.g., security issues and router or node design) mentioned in the CCN proposal [30], however, it pays less attention to the caching operations which are the major concern of all ICN architectures. ccnSim is based on Omnet++ [158] and is developed primarily to focus on the caching capabilities of the architectures proposed in [30]. Although ccnSim has the support for simulating large catalog and can handle millions of content requests, the main limitation of this simulator is that it lacks to execute the trace-driven simulations, which is one of the most vital attributes for in-network caching [159]. Therefore, it is not considered the appropriate simulator for testing the performance of in-network caching. ndnSim is another open source simulator based on NS-3 [160] designed for NDN architecture. Although ndnSim provides supports for all the required modules to analyze the transport behavior of the ICN architecture (even though it provides support for router caching), but its structure is not appropriate for the evaluation of in-network caching strategies [159]. Like ccnSim, ICN Simulator is also designed in Omnet++ for PAL project [161]. This simulator is GUI-based, therefore, very easy to configure any topology in it and the results can be seen instantaneously. The simulator provides support for the following modules: 1) the Publish-Subscribe Router, 2) the Publisher, 3) the Subscriber, 4) the Information Item Table, and 5) the Topology Manager, but it does not support caching at all.

3.5.4 Features of SocialCCNSim

Cesar Bernardini developed SocialCCNSim [142], based on SONETOR [162], which is a set of utilities that generates synthetic social network traces. These social network traces represent interaction of users in a social network fashion or in a regular client-server fashion. Any caching strategy can be implemented in SocialCCNSim, but here we provide a list of strategies that have been evaluated in it [142], i.e., Leave Copy Everywhere (LCE) [30], Leave Copy Down (LCD) [32], ProbCache [41], Cache 'Less For More' [106], MAGIC [163], Leave Copy on the Edge, Rand Copy One, Most Popular Content (MPC) [33], and Betweenness-Centrality [164], and two replacement policies, i.e., Least Recently Used (LRU) and Least Frequently Used (LFU). SocialCCNSim supports two *Social Network* topologies, i.e., Facebook [165, 166, 167] and LastFM [168, 169], and the following *Network Topologies*: Abilene, GEANT, DTelekom, Tree, Tiger, and Diamond. Apart from all the above mentioned benefits, SocilaCCNSim also supports users' mobility, which is a critical issue and great concern of the future Internet.

3.5.5 OPNET Modeler

OPNET is the most scalable commercially available and fastest software that normally takes a few minutes to complete simulations [170, 138]. It has an enormous user community, i.e., more than 500 companies, universities, and service providers use OPNET throughout the world. OPNET builds a virtual network that includes devices, for example, routers, switches, hubs, servers, and PCs; it also provides protocols and application software. To know the basic ideas of networking, OPNET facilitates and equips the students to efficiently manage and troubleshoot the infrastructures of real-world networks [171]. OPNET provides support for simulating any kind of network that exists in the world, for example, Local Area Networks (LANs), Wide Area Networks (WANs), and the future Internet, i.e., ICN.

3.5.6 Simulation Steps

It is helpful to divide the simulation performance evaluation into steps. The performance evaluation has been divided into either eight or twelve steps by earlier researchers. For instance, Hassan and Jain [172] have decomposed the simulation task into eight steps, as shown in Figure 3.9 [133, 172]. Their work has two stages, i.e., Pre-software stage and software stage, which was followed throughout this research.

A. Pre-software stage

This stage consists of the following four steps:

1. research objectives are explained accurately,
2. network model is created and fixed parameters are chosen: in this step network topology is drawn on a piece of paper and suitable network parameters are cho-

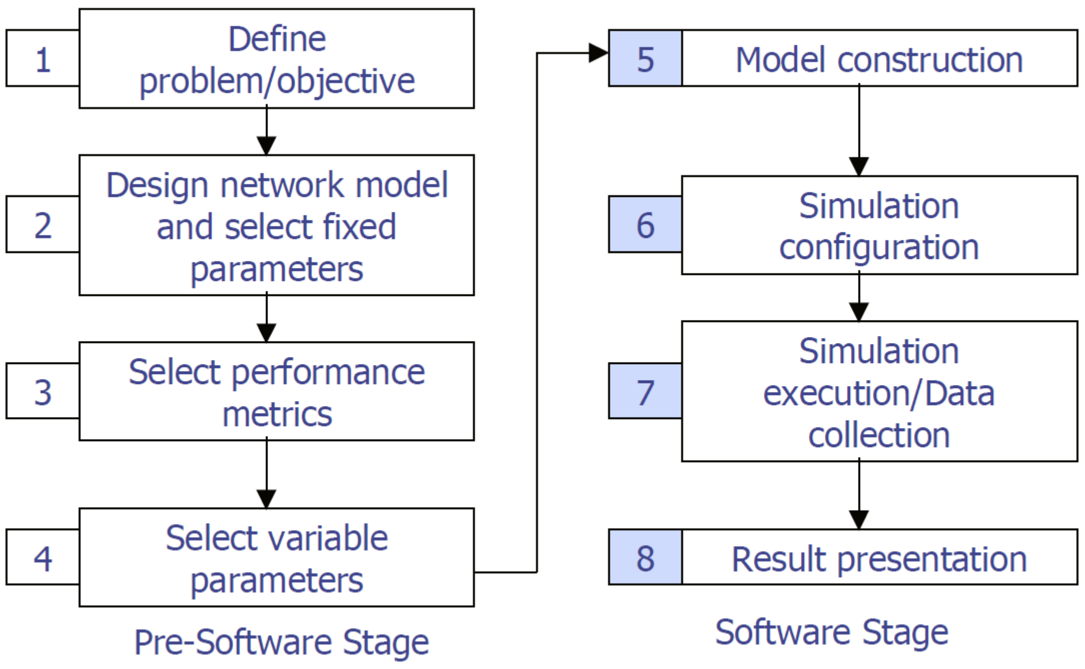

1
Define problem/objective
2
Design network model and select fixed parameters
3
Select performance metrics
4
Select variable parameters
Pre-Software Stage
5
Model construction
6
Simulation configuration
7
Simulation execution/Data collection
8
Result presentation
Software Stage

Figure 3.9. Simulation Steps [172]

sen to reflect an acceptable and real scenario,

3. performance metrics are selected for the evaluation of proposed strategy,
4. variable parameters are selected: generally, in ICN cache management, the purpose of performance evaluation is to examine the effect of particular variables on the chosen performance metrics.

B. Software stage

This stage consists of the following four stages:

1. network topology designed in step A-2 is implemented in a simulator,
2. simulator is configured in such a way that appropriate performance metrics are selected in step A-3,
3. simulator is executed and when the simulation is finished, the performance metrics data are collected, and

Table 3.1
Simulation Scenario

Parameter	**Description**
Cache Size	100 - 1,000 elements
Catalog Size	10^6 elements
α Parameter	0.7, 1.0, 1.5, 2.0
Topology	Abilene, Tiger, GEANT, DTelekom
Social Network Topology	Facebook [165, 166, 167]
Simulator	SocialCCNSim [142, 173]
No. of Simulation	10 runs

4. the collected data in step B-3 are presented in a meaningful form, and the presented results are then interpreted.

In this study, the performance metrics are cache hit rate, redundancy, content retrieval delay, memory utilization, and stretch ratio. In addition, four ISP-level topologies were selected for the fair evaluation, i.e., Abilene, GEANT, Tiger, and DTelekom. In the final stage, the FlexPop evaluation was done using simulation, where the selected parameters were cache size, catalog size, network topology, Zipf probability model, and simulation time. The performance of FlexPop was then compared and analyzed with the results of three other caching strateies, i.e., LCE, ProbCache, and MPC.

3.5.7 Simulation Scenario

The proposed work has been simulated in the SocialCCNSim and OPNET Modeler 18.0. OPNET Modeler is an old simulation tool that has been tested by hundreds of thousands of researchers and academicians, however, SocialCCNSim is a new simulator designed only for testing the performance of ICN caching, therefore, its validation is indispensable.

In our simulations, SocialCCNSim is used for evaluating four caching strategies (i.e., LCE, LCD, ProbCache, and Betweenness-Centrality) with LRU cache replacement policy as in [58] for all ICN nodes. The LRU policy is chosen for replacement because it presented the best results, as submitted in [5]. In SocialCCNSim, for modeling content requests, the Zipf distribution function [174, 175] is used. It is mentioned earlier that six network topologies are included in the simulator and every caching strategy can be tested on any of the mentioned topologies in the SocialCCNSim, but we have used four ISP-level topologies, i.e., Abilene, Tiger, GEANT, and DTelekom with Facebook [165, 166, 167] for evaluating these strategies (see Table 3.1). In our simulations, the Zipf probability distribution is used as popularity model with the α parameter varies between 0.7 and 2.0; the cache size (which specifies the available space in every node for temporally storing content objects) ranges from 100 to 1,000 elements (1GB to 10 GB); and the catalog (which represents the total number of contents in a network) is 10^6 elements. The scenario was simulated for 10 number of runs and the average values were taken for cache hit rate, as shown in Figures 3.10 to 3.13, whereas their numerical result is presented in Table 3.2. The simulations were performed for 10 times because the evaluated cache size is 1GB to 10GB, therefore for each cache size the simulation was run and the average result was collected for all metrics.

3.5.8 Performance Metrics

Different researchers think of performance metrics different things depend upon the context in which they are used. Because there is no standard to select the metrics for the evaluation of caching efficiency, several metrics have been used in the literature. Among these metrics the most common are Server Load, Bandwidth Utilization, Cache Hit Rate, Content Eviction Rate, Cached Element Rate, Diversity, Content Retrieval Delay, Memory Utilization, Redundancy, Stretch Ratio, and Cache Miss Rate. To sum up, in all studied metrics in the literature, Server Load, Bandwidth Utilization, Content Eviction Rate, Cached Element Rate, and Cache Miss Rate are restricted

Table 3.2
Hit Rate on Different Topologies

Topology	**Popularity model α**				**Cache Size**		**Cache Hit**			
	0.7	**1.0**	**1.5**	**2.0**	**100**	**1,000**	**LCE**	**LCD**	**Prob-Cache**	**Centrality**
Abilene	√	-	-	-	√	-	10	11	12	8
	√	-	-	-	-	√	15	20	18	14
	-	√	-	-	√	-	13	17	14	9
	-	√	-	-	-	√	29	34	32	23
	-	-	√	-	√	-	20	23	22	18
	-	-	√	-	-	√	40	46	44	35
	-	-	-	√	√	-	25	32	35	26
	-	-	-	√	-	√	52	59	65	52
Tiger	√	-	-	-	√	-	11	12	13	9
	√	-	-	-	-	√	17	23	21	15
	-	√	-	-	√	-	18	20	21	11
	-	√	-	-	-	√	33	38	38	27
	-	-	√	-	√	-	25	24	27	19
	-	-	√	-	-	√	44	51	54	39
	-	-	-	√	√	-	31	37	35	30
	-	-	-	√	-	√	64	68	72	60
GEANT	√	-	-	-	√	-	11	12	13	9
	√	-	-	-	-	√	17	23	21	15
	-	√	-	-	√	-	20	21	22	12
	-	√	-	-	-	√	34	39	39	28
	-	-	√	-	√	-	25	24	28	20
	-	-	√	-	-	√	44	51	54	40
	-	-	-	√	√	-	32	37	36	31
	-	-	-	√	-	√	65	68	72	60
DTelekom	√	-	-	-	√	-	13	13	14	9
	√	-	-	-	-	√	19	24	23	15
	-	√	-	-	√	-	23	26	27	19
	-	√	-	-	-	√	38	42	45	36
	-	-	√	-	√	-	31	35	40	29
	-	-	√	-	-	√	55	59	64	49
	-	-	-	√	√	-	38	39	44	35
	-	-	-	√	-	√	68	71	77	65

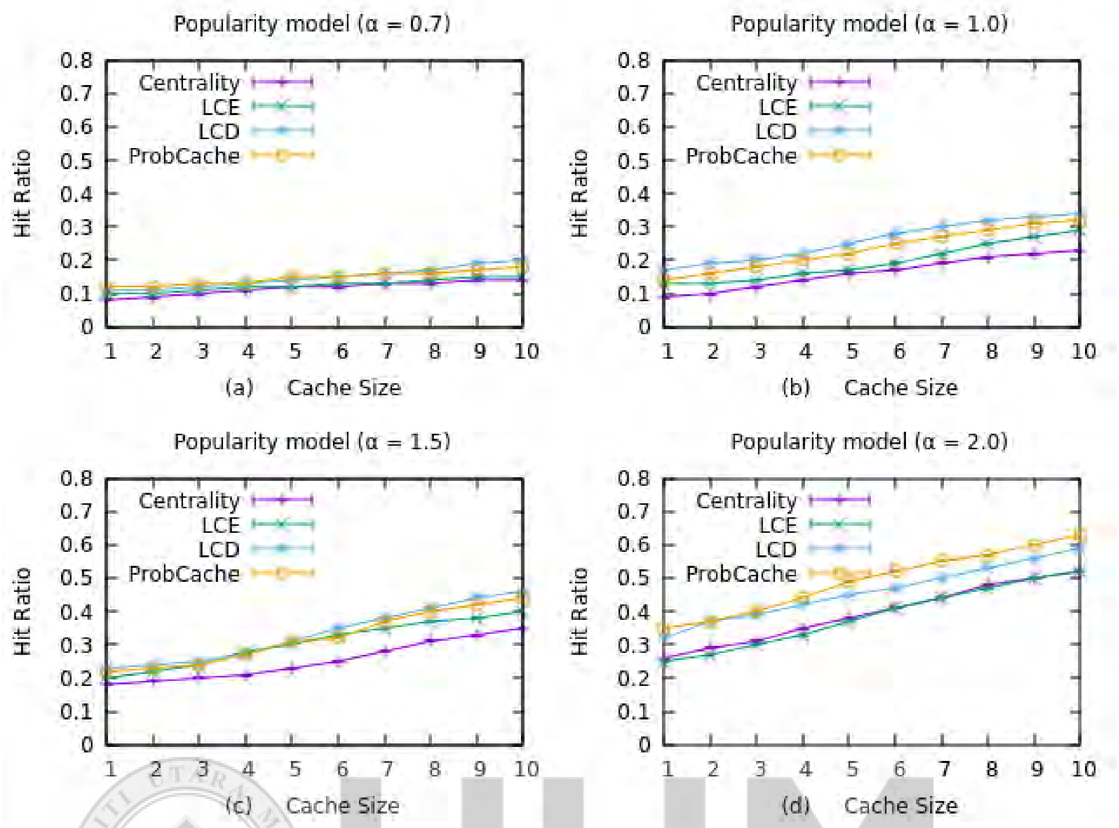


Figure 3.10. Cache Hit on Abilene Topology

to partial information. Similarly, Diversity shows that how many different kinds of contents are cached in the content store. As our proposed strategy is designed for multimedia applications, we consider that diversity does not provide better intuition. In addition, Cache Hit Rate combines the properties of Content Eviction Rate, Cache Miss Rate, and Cached Element Rate, therefore, it is selected for the fair evaluation in this thesis. Other metrics include: Redundancy, Content Retrieval Delay, Memory Utilization, and Stretch Ratio.

Cache Hit Rate: Hit rate means the portion of content requests satisfied by caches implemented within the network. This metric shows the potential of the caching strategy to moderate the number of redundant content copies.

Redundancy: In the context of ICN, redundancy means the number of content copies

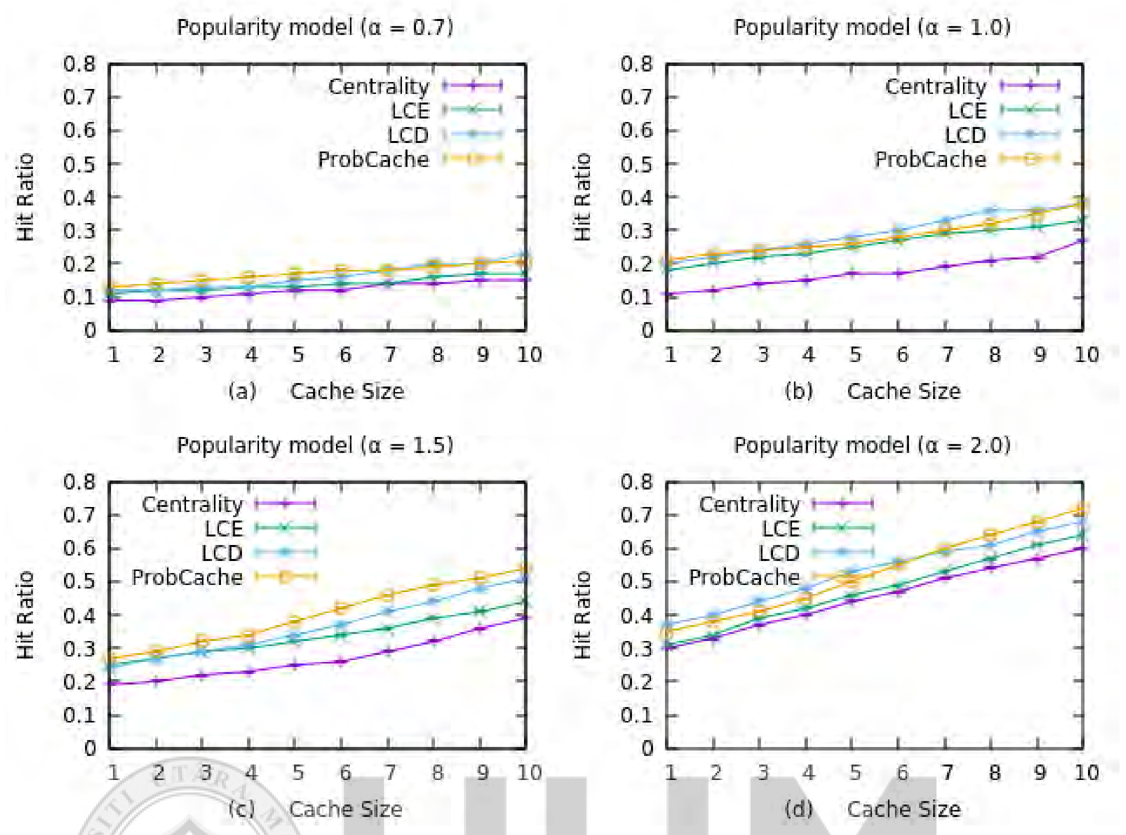


Figure 3.11. Cache Hit on Tiger Topology

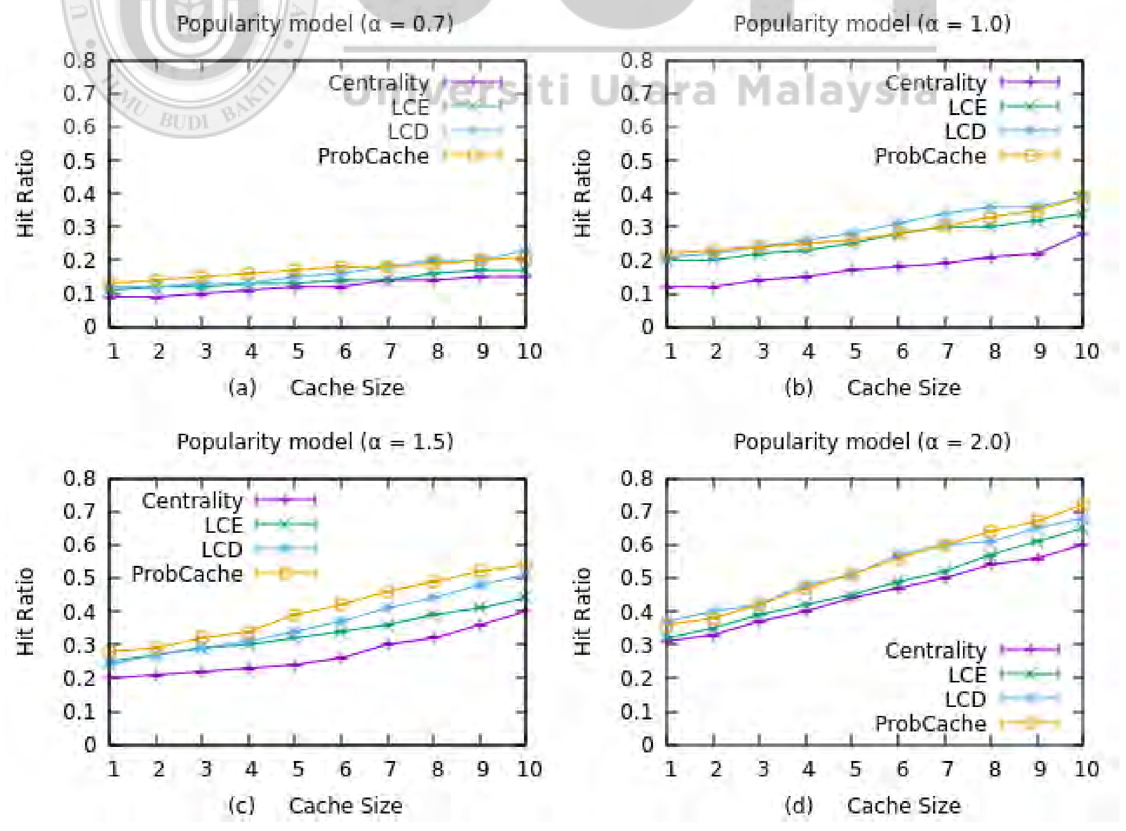


Figure 3.12. Cache Hit on GEANT Topology

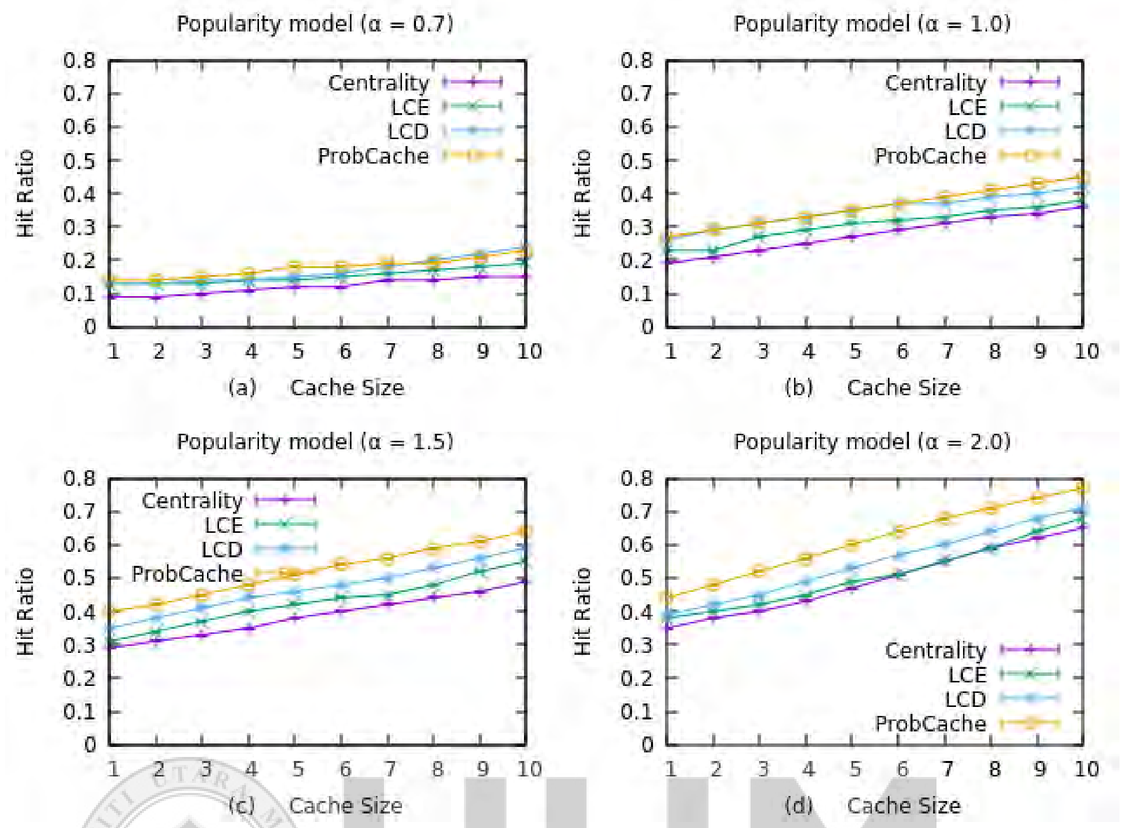


Figure 3.13. Cache Hit on DTelekom Topology

cached at more than one location inside the network. In our simulation, path redundancy is considered rather than content redundancy. In other words, only the ratio of redundant contents are computed which are cached along the publisher to subscriber path rather than the cached contents in the whole network.

Content Retrieval Delay: It is the time between a request from a source node and reply from the server (or the node having a copy of the object for which request received) back to the source.

Memory Consumption: If a content item is stored at more than one location, then maximum memory is utilized and there is no space for the new arriving contents. Maximum memory consumption can also affect the bandwidth utilization such that if the memory is full then the new arrived contents will replace the popular cached ones.

Hence, the subsequent requests for a popular content will not find it locally, thus, all those requests will be forwarded to the server(s) and therefore maximum bandwidth will be utilized.

Stretch: It refers to the number of hops a content request travels from client to server to find a content in the network. Stretch depends on *hop decrement*, that is, if the *hop decrement* is high the content delay will be reduced and vice versa. The high *hop decrement* means that the content will be cached close to the subscriber, therefore, it will not take long time in downloading.

A detailed explanation about the performance metrics is given in Chapter Five.

3.6 Summary

This chapter comprehensively explained the approach to guarantee the achievement of research objectives. The research focuses on designing FlexPop: popularity based caching strategy for ICN. Here, four major activities of the research, in line with DRM, were outlined. Research Clarification (RC) stage is the first activity, which presents techniques to support the preliminary phase of this research. The purpose of RC is to categorize and sort out a research problem, research questions, and objectives which have both academic and practical significance. Descriptive Study-I (DS-I) is the second phase, which discusses steps for obtaining adequate knowledge of the current situation. This phase includes designing a reference model as well as proposing a conceptual model. Prescriptive Study (PS) is the third phase which focuses on methods adopted in designing the proposed cache management strategy. DS-II is the last phase that discusses the evaluation of the designed FlexPop caching strategy. This section also includes validation of the simulator and describes which simulation steps are followed for the performance evaluation. The performance of four strategies: LCE, LCD, ProbCache, and Betweenness-Centrality, have been compared in terms of cache hit rate. Next chapter discusses the proposed system model for the FlexPop strategy.

CHAPTER FOUR
FLEXPOP: A POPULARITY-BASED CACHING STRATEGY

This chapter proposes a new strategy, named FlexPop, for caching multimedia application in ICN. The proposed strategy focuses on content placement and eviction, and determines *what and where to cache*. Section 4.1 introduces the importance of in-network content placement jointly considered with some challenges faced during ICN caching and then discusses the two caching complication, i.e., cache deployment and content placement. Section 4.2 explains system model with the help of a fuzzy approach for content caching. The need for a flexible popularity-based caching strategy and a content eviction approach are described in Section 4.3. The FlexPop's parameters selection along with verification and validation are presented in Section 4.4, while Section 4.5 concludes the chapter.

4.1 Caching in ICN

The increasing demand for multimedia contents requires more convenient techniques for content retrieval. ICN is such an approach that satisfies this demand by offering named content and designating in-network caching [17]. In ICNs, a content store (CS) with the potential of in-network caching can store some content items for future use [30]. In-network caching can significantly reduce service load on the servers, content retrieval delay, and traffic in the network [175, 110]. To cope with in-network caches in ICNs, two critical challenges need to be jointly considered. One is the content routing that decides where to forward content requests and how to deliver contents, and the other is caching policy that chooses which content item should be stored in CS.

Unlike the current Internet architecture that was designed for source to destination communication, customers are only interested in actual content rather than its location.

For this reason, novel ICN architectures such as Publish-Subscribe Internet Technology (PURSUIT) [25, 23], Publish-Subscribe Internet Routing Paradigm (PSIRP) [22], Network of Information (NetInf) [176] and CCN [30] have been defined which give high priority to effective content dissemination at large scale. Among all these novel architectures, the research community has been considerably attracted by CCN [21]. CCN is an architecture that efficiently distributes popular contents to a possible number of clients [30, 64]. A fundamental characteristic of CCN is cache management with caching strategies, which determines what to cache and where.

The major complication in ICN is the cache deployment and content placement [177, 178, 179, 180]. These issues are presented in the following subsections.

4.1.1 Cache Deployment

Cache deployment is the way of selecting the proper position of caching node such as router on the path from publisher to subscriber. The deployment of cache routers may drastically effect the performance of overall ICN network in terms of cache hit rate and content retrieval delay [181]. More specifically, redundancy elimination achieved substantial attention from the research community both in the current Internet [182, 183, 184, 185, 186] as well as in ICN [44, 187, 188, 189, 190]. If the caching node is deployed in a proper location, the network bandwidth is efficiently utilized. For instance, if a caching router is placed in such a position that maximum content requests pass through it, that will satisfy those requests locally rather than downloading contents from the main server.

4.1.2 Content Placement

To store contents, for example, videos, images, or web documents for some time, an approach is used which is known as caching. The key goals of caching contents are to reduce the utilization of bandwidth and minimize the server overhead as well as content retrieval delay [191]. In the early 1990s, the concept of web cache was introduced by some companies for accessing the Internet through a proxy server (HTTP server) [58]. A proxy server is used to forward requests to the remote servers and receive responses from servers and forward back to the intended destinations [192]. The proxy server consists of a temporary memory, which is known as cache. The main issue among caches is their coordination. For this reason the research community has proposed some cache management solutions, for example, distributed caches and hierarchical caches [58].

In distributed caching, the information is stored at the edge routers (especially where the congestion is minimum) to reduce content retrieval delay [193]. In hierarchical caching, the cache is deployed at every level of the network, i.e., local, institutional, regional, and Internet Service Provider (ISP) in such a fashion that requests travel through the hierarchy of caches [194, 195]. Hierarchical caching did not gain momentum, however, with the introduction of CDNs, distributed caching gained popularity, in which the contents are stored at the edge routers near to the customers. Almost 20% of all web traffic is delivered by the Akamai CDN [196], while by 2016, the expectation is 50% of the worldwide traffic [58]. The Akamai network consists of more than 61,000 servers [196], but every device is not used to store contents in this solution of web caching. On the other hand, in-network caching is proposed in ICN architecture, where every network node has the capabilities of caching contents. However, as in [197], for storing contents there will exist at least 3.7 billion of available caches. And hence, the deployment of caches, i.e., placement of contents is a challenging and yet to resolve issue in ICN.

4.2 System Model

In this section the fuzzy *Multiple Attribute Decision Making* (MADM) method [135] is discussed, which is used to resolve the problems associated with fuzzy data. In other words, it is used for the measurement of values of different attributes in different units [135].

In the context of ICN, MADM counts the popularity values of incoming contents to be recommended for caching. In the following subsection we discuss an approach for content caching based on fuzzy MADM.

4.2.1 Fuzzy MADM Approach for Content Caching

Fuzzy set theory [198], which is a perfect means for modeling uncertain (imprecise) data, is the basis for MADM approach. MADM refers to making selections among some courses of action in the presence of multiple, usually conflicting, attributes [135]. In our proposed strategy, the contents are fuzzy because of its unavailability. In a fuzzy MADM problem, in ICN caching, the popularity values of contents can be represented by fuzzy sets. Detailed explanation about fuzzy set can be found in [135].

There are many MADM techniques, but the first part of our proposed strategy, i.e., Content Placement Mechanism (CPM), is based on SAW method in which the total values of an alternative are calculated and the one with the higher score is selected. The SAW procedure is given as follows [199, 200, 135]:

Suppose we have contents C_j, $j = 1, ..., n$, and their values $\omega = \omega_1, \cdots, \omega_n$, the popularity of content P_i, is calculated as:

$$U_i = \frac{\sum_{j=1}^{n} \omega_j \rho_{ij}}{\sum_{j=1}^{n} \omega_j} \tag{4.1}$$

where ρ_{ij} is the outcome of the i^{th} content about the j^{th} value (popularity). This is the simplest form in Multiple Attribute Utility Theory (MAUT). The most popular content, P^*, is then selected such that

$$P^* = P_i | max_i U_i \tag{4.2}$$

if both ω_j and ρ_{ij} are fuzzy sets defined as:

$\omega_j = \{(x_j, \mu\omega_j(x_j))\}, \forall j$

and

$\rho_{ij} = \{(y_{ij}, \mu\rho_{ij}(y_{ij}))\}, \forall i, j,$

where $\mu\omega_j(x_j)$ and $\mu\rho_{ij}(y_{ij})$ take values and x_j and y_{ij} take the actual numbers on the real line $\mathbb{R}$, the popularity of content P_i, $U_i = \{(\nu_i, \mu U_i(\nu_i))\}$, can be calculated as follows

$$\nu_i = \frac{\sum_{j=1}^{n} x_j y_{ij}}{\sum_{j=1}^{n} x_j} \tag{4.3}$$

$\mu U_i(\nu_i)$, which is the membership function, can be calculated as

$$\mu U_i(\nu_i) = sup_u\{[\prod_{j=1}^{n} \mu\omega_j(x_j)] \prod [\prod_{j=1}^{n} \mu\rho_{ij}(y_{ij})]\} \tag{4.4}$$

where $u = (x_1, \cdots, x_n, y_{i1}, \cdots, y_{in})$.

However, the function $\mu U_i(v_i)$ cannot be obtained directly if $\mu\omega_j(x_j)$ and $\mu\rho_{ij}(y_{ij})$ are continuously differentiable functions.

To solve this problem, Kwakernaak's approach is used as follows [135, 201]:

Given fuzzy weights $\omega_j = \{(x_j, \mu\omega_j(x_j))\}$ and fuzzy attribute $\rho_{ij} = \{(y_{ij}, \mu\rho_{ij}(y_{ij}))\}$ for the popularity of content P_i, to derive $U_i = \{(v_i, \mu U_i(v_i))\}$, the following steps are used:

step 1: Choose an α_0 level.

step 2: For content P_i, determine the following real numbers:

$$y'_{ij} = min\{y_{ij} \varepsilon R | \mu\rho_{ij}(y_{ij}) \geqslant \alpha_0\}, \forall j, \tag{4.5}$$

$$y^*_{ij} = max\{y_{ij} \varepsilon R | \mu\rho_{ij}(y_{ij}) \geqslant \alpha_0\}, \forall j, \tag{4.6}$$

$$x'_j = min\{x_j \varepsilon R | \mu\omega_j(x_j) \geqslant \alpha_0\}, \forall j, \tag{4.7}$$

$$x^*_j = max\{x_j \varepsilon R | \mu\omega_j(x_j) \geqslant \alpha_0\}, \forall j, \tag{4.8}$$

Step 3: The ρ_{ij}, which is the outcome of the i^{th} content about the j^{th} popularity value, can be represented, at the α_0 level, by $[y'_{ij}, y^*_{ij}]$.

Put $y'_{ij}, \forall j$, and $x'_j, \forall j$, in such an order that

$$\kappa'_1 \leq \kappa'_2 \leq \cdots \leq \kappa'_n \tag{4.9}$$

and

$$\ell'_1 \leq \ell'_2 \leq \cdots \leq \ell'_n \tag{4.10}$$

where $\kappa'_1 = min_j y'_{ij}$ and $\kappa'_n = max_j y'_{ij}$,

while $\ell'_1 = min_j x'_j$ and $\ell'_n = max_j x'_j$.

In the same fashion, y^*_{ij} and x^*_i are recognized in such an order that

$$\kappa^*_1 \leq \kappa^*_2 \leq \cdots \leq \kappa^*_n \tag{4.11}$$

and

$$\ell^*_1 \leq \ell^*_2 \leq \cdots \leq \ell^*_n \tag{4.12}$$

where $\kappa_1^* = min_j y_{ij}^*$ and $\kappa_n^* = max_j y_{ij}^*$,

while $\ell_1^* = min_j x_j^*$ and $\ell_n^* = max_j x_j^*$.

Step 4: Let U_i at the α_0 level be $C\alpha_0 = [a\alpha_0, b\alpha_0]$,

where $a\alpha_0$ and $b\alpha_0$ are the lower and upper bound, respectively.

$a\alpha_0$ and $b\alpha_0$ are defined as:

$$a\alpha_0 = min_j[\frac{\sum_{r=1}^{j} \ell_r^* \kappa_r^{'} + \sum_{r=j+1}^{n} \ell_r^{'} \kappa_r^{'}}{\sum_{r=1}^{j} \ell_r^* + \sum_{r=j+1}^{n} \ell_r^{'}}], 0 \leq j \leq n, \tag{4.13}$$

and

$$b\alpha_0 = max_j[\frac{\sum_{r=1}^{j} \ell_r^{'} \kappa_r^* + \sum_{r=j+1}^{n} \ell_r^* \kappa_r^*}{\sum_{r=1}^{j} \ell_r^{'} + \sum_{r=j+1}^{n} \ell_r^*}], 0 \leq j \leq n. \tag{4.14}$$

$a\alpha_0$ can be computed according to Equation 4.3, where $y_{ij}, \forall j$, take $y_{ij}^{'}$ as their values, then the minimum of v_i in Equation 4.3 is guaranteed.

As in our proposed strategy the contents are cached based on popularity, a popular content P^*, as in [135, 202], can be cached according to the following formula:

$$D_i = \sqrt{\sum_{j=1}^{n} \omega_j^2 (\rho_{ij} - S_j)^2}, i = 1, 2, \cdots, m. \tag{4.15}$$

where ρ_{ij} is the outcome of the i^{th} content about the j^{th} popularity value, S_j is the size of content store where contents are cached, and ω_j is the popularity value of the j^{th} content.

4.3 The Need for a Flexible Caching Strategy

Generally, most ICN studies focus on cache replacement policies (for example LRU and LFU) [203, 204, 205] over various topologies such as educational ISP [116] and binary trees [174]. Relating to caching strategies, some work demonstrates that, to achieve high caching performances, huge cache memory is required [175] while the other [106] opposes that storing less content can attain similar results by only placing contents in a subset of routers along the delivery path. It is therefore necessary to design flexible caching strategy to achieve better performance in terms of cache hit rate, redundancy, content retrieval delay, stretch, and memory utilization.

Content placement and eviction schemes play a vital role in ICN's overall performance. The LFU and LRU replacement policies were mainly proposed for ICN, but these policies suffer from low efficiency, i.e., LRU uses a time stamp of the content while LFU calculates distribution frequency of the content. In both policies, content popularity is ignored, which causes inaccuracy in recognizing the degree of a newly incoming content. As a result, insufficient content replacement may happen, i.e., new unpopular contents may replace the old popular ones.

In order to tackle the above inadequacy and enhance the accuracy of content replacement decision, various caching strategies have been developed (discussed in Chapter Two). Most of them focus on cache hit rate and redundancy, however, memory utilization, stretch, and content retrieval delay, which are the essential metrics in measuring the performance of ICN, are mostly ignored. In this research a new caching strategy, named Flexible Popularity-based Caching (FlexPop), is proposed to overcome the above critical issues.

4.3.1 FlexPop Strategy

The FlexPop strategy consists of two mechanisms, i.e., 1) Content Placement Mechanism (CPM), which is responsible for content caching (see Figure 4.1), and 2) Content Eviction Mechanism (CEM), as discussed in Section 4.3.3, deals with the eviction of contents in case the memory of content store is full and there is no room for the incoming content to be cached. In FlexPop, each ICN router maintains a table called *Popularity Table* (PT) containing information concerning the popularity of a content name and that is in the form of a content counter along with a popularity tag. Indeed, FlexPop records popular content by locally counting the number of incoming requests for a content.

When a request for a new content arrives, it is stored in the PT. This table has a counter called *Content Popularity* (C_p) that defines the popularity value of a content coming from upstream or forwarding downstream. In case the popularity value of a content is greater than or equal to 2, it is forwarded to *Comparison Table* (CT), otherwise remains in PT. The CT contains information about the content names forwarded to *Content Store* (CS). A counter *Popularity Value* (V_p) in the CT checks the popularity of a content request, if it is greater than the popularity value of the stored contents then it is recommended for caching in the CS. The CT, which has less content names than PT, is shared with neighbor routers.

As the popularity value of a content reaches the popularity threshold, which is greater than or equal to 2, it is stored on the node having maximum outgoing interfaces, as in [106]. The lower bound of the popularity is set to 2 because if a content is requested

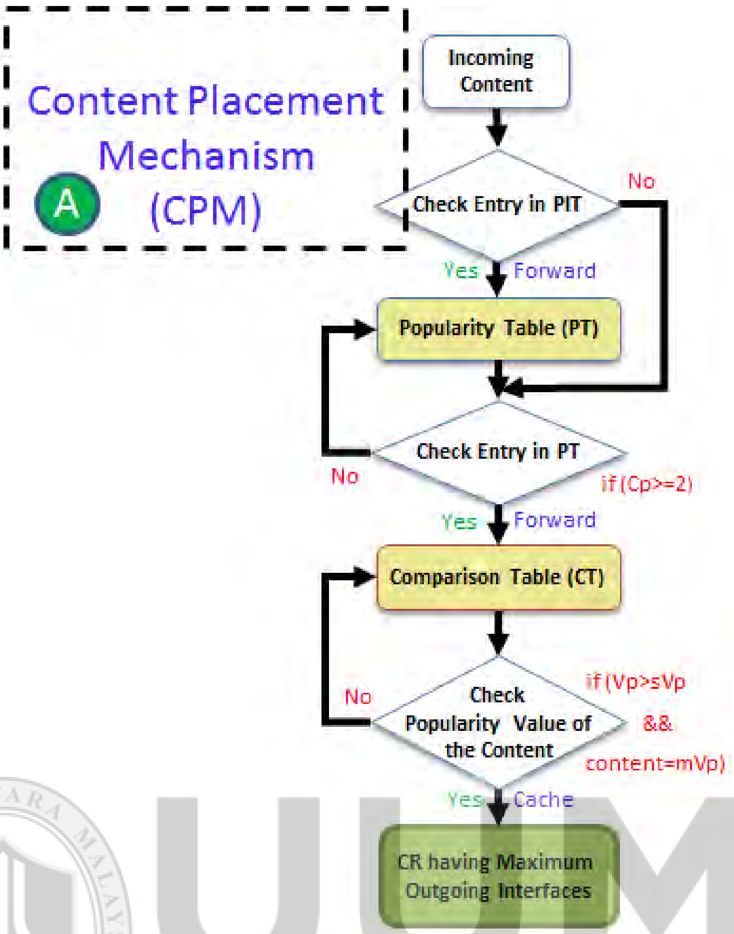


Figure 4.1. Content Placement Model

two times then it is recommended for caching. Here, the lower bound is 2 because to filter popular contents from unpopular ones, as all unpopular contents will have popularity value 1. If the popularity is set to a higher value then it can cause downstream latency and slow convergence of hit rate. On the other hand, caching every incoming content, even if memory is not full, can increase searching overhead during replacement process.

As in Algorithm 4.1, if a request arrives, it is first checked in the *Pending Interest Table* (PIT) and then forwarded to PT. The PT contains the requested content ID, name, and its popularity value. If the popularity value of a content reaches its minimum value, i.e., if $Cp >= 2$, it is forwarded to CT. In contrast to PT, CT has only popular content items, therefore, this table is shared with all neighbor routers. In CT, the counter Vp

Algorithm 4.1 Content Placement Mechanism (CPM)

```
Let $\aleph$ represent Network:
Incoming content = $I_x$;
$If(\exists\, I_{xi}\ in\ PIT,\ \forall I_x)$
  Forward $I_{xi}$ to $PT$;
$Else$
  Cache $I_{xi}$ in $PT$;
$End\ If$
$Check\ \ I_x\ in\ PT$:
$If(C_p \geqslant 2, \forall I_{xj})$
  Forward $I_{xj}$ to $CT$;
$Else$
  Leave $I_{xj}$ in $PT$;
$End\ If$
  Recommend $CT \Rightarrow NR$;
[where $NR$ is the neighbor router]
$xCT \subseteq NR;$
$Check\ V_p, \forall I_x$:
$If\ (V_p > sV_p,\ \&\&\ I_{xk} = mV_p)$
  Cache $I_{xk} = CR_{max}$;
[where $CR_{max}$ is the router having maximum outgoing interfaces.]
$Else$
  Leave $I_{xk}$ in $CT$;
[where $i, j, k = \{1, 2, \cdots, n\}$, $i \neq j \neq k$.]
$End\ If$
```

will check the value of content, if $Vp > sVp$ and $content = mVp$, then it is forwarded to CS. Here, CS means that router which has maximum outgoing interfaces, in this case router R2 in Figure 4.2.

Here a question arises that if the popularity value of content should never be less than 2 in CT then what is the need for checking content popularity here?

This check is needed because if, for example, there are three content items in CT and all have different popularity value (definitely greater than 2), then to decide which content item should be forwarded to CS if cache is full. In this case, that content item which has the maximum popularity value in CT (i.e., $content = mVp$) and the value of this content is greater than the value of stored one (selected for replacement by LRU)

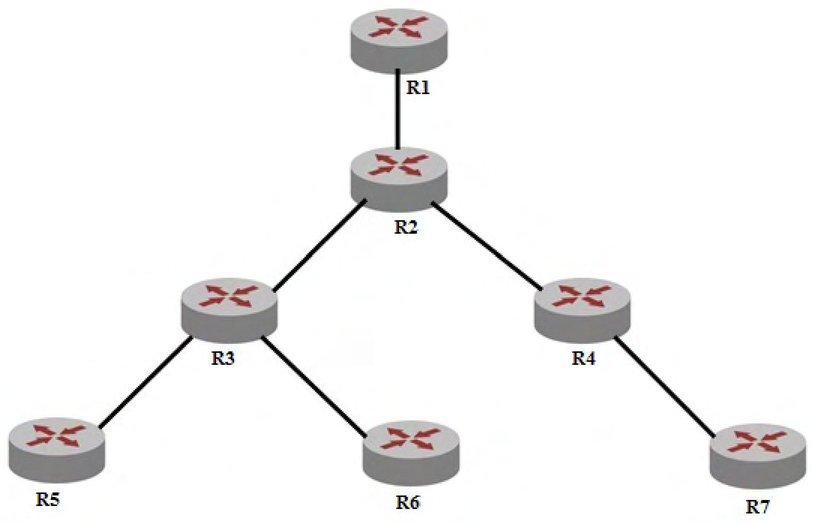


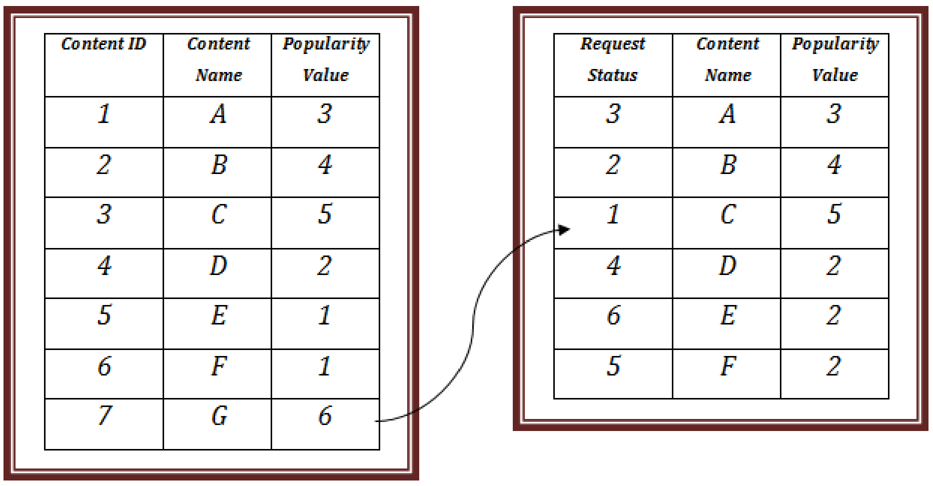
Figure 4.2. A Network Topology

in CS (i.e., $Vp > sVp$) will be forwarded for caching.

4.3.2 The Role of Sharing CT with Neighbor Routers

Here it is necessary to explain the purpose of sharing CT with neighbor routers. To know this, consider the topology in Figure 4.2. If a user X (connected to router R5) requests content B and the popularity value of B in its CT has reached 3 (see Figure 3.4 in Chapter Three), router R5 will share this CT with router R3 and in turn R3 will share it with its neighbor router R6. Furthermore, consider another user Y is connected to router R6 who requests the same content B. With this request, the popularity value of B will be updated in CT that will become 4 and hence, the CT will be shared with the neighbor router R3, and R3 will update its CT accordingly. As routers R2 and R5 are neighbors to router R3, the same CT will be shared with them and they will also update their CTs accordingly.

On the other hand, if router R5 does not share CT with its neighbor, then router R6 (who has received request for content B) will have the popularity value being 1 for B, even though its popularity value has already reached 3 at router R5 and has been rec-



Content ID	Content Name	Popularity Value
1	A	3
2	B	4
3	C	5
4	D	2
5	E	1
6	F	1
7	G	6

Request Status	Content Name	Popularity Value
3	A	3
2	B	4
1	C	5
4	D	2
6	E	2
5	F	2

Figure 4.3. An Example of Popular Content Replacement

ommended for caching. Moreover, Router R6 will know from the CT (received from router R3) that content B is cached by router R2 (which has the maximum outgoing interfaces) and therefore will forward the request to router R2 instead of forwarding to the server. As a result, this will reduce the downstream latency and expedite the access time.

4.3.3 An Expected Case

If V_p is set to a higher value then some popular contents will not contribute in the cache hit rate. On the other hand, if V_p is kept very low, e.g., 1, then cache may be overflowed and hence some unpopular contents will replace popular ones during replacement process.

As in [114], content items are stored in cache as long as space is available, so when, for example, a new content item arrives then according to the nature of LRU there is a possibility of evicting the most popular content by a least popular one. As shown in Figure 4.3, the popularity of content C at time t_1 and time t_2 is 5. Similarly, the popularity of E and F at time t_1 is 1 but at time t_2 their popularity is 2. Here, the popularity of C is higher than E and F but it is the least recently used, therefore, it will

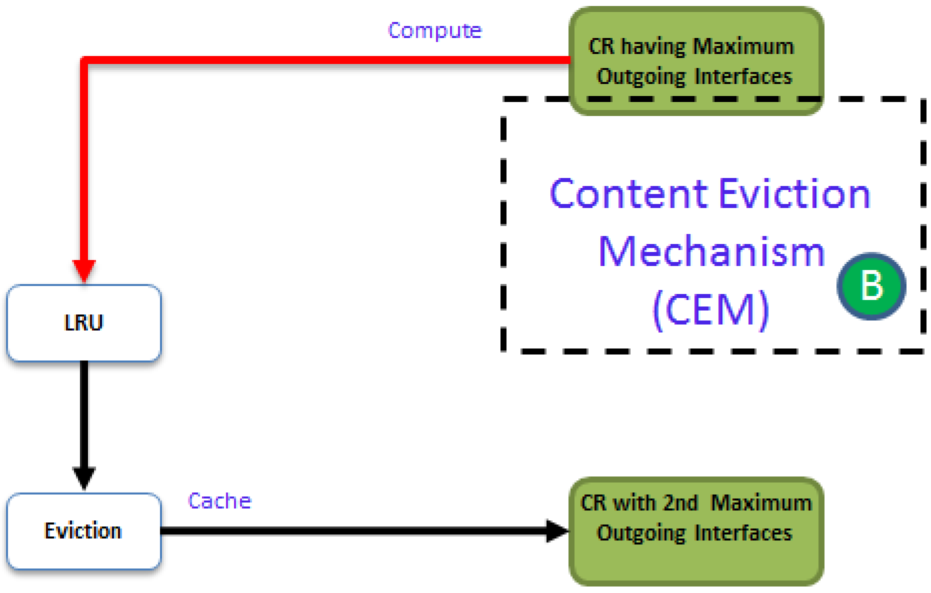

Compute
CR having Maximum Outgoing Interfaces
Content Eviction Mechanism (CEM)
B
LRU
Eviction
Cache
CR with 2nd Maximum Outgoing Interfaces

Figure 4.4. Content Eviction Model

Algorithm 4.2 Content Eviction Mechanism (CEM)

```
$While(CR_{max} = !0)$:
    Check $I_{xi}$ against $I_{xj}$;
$If(\exists\, I_{xi}\ \&\&\ I_{xj})$
$I_{xi}\ \&\&\ I_{xj} \leftarrow LRU\ \ in\ CR_{max}$;
    Evict $I_{xi}|I_{xj}$ from $CR_{max}$;
    Cache $I_{xi}$ in $CR_{max2}$;
$End\ If$
[where $CR_{max2}$ is the router having $2^{nd}$ highest outgoing interfaces.]
[where $i, j = \{1, 2, \cdots, n\}$ , $i \neq j$.]
```

be replaced by the new popular content G whose popularity is 6.

In FlexPop, when the cache of a router becomes full and a new popular content needs to be stored, CEM will replace the content in cache according to LRU. As FlexPop stores popular contents which may possibly be evicted by LRU, the evicted content will be recommended to a router for caching who has the second highest number of outgoing interfaces (R3 in Figure 4.2). The workflow of CEM is presented in Figure 4.4 and Algorithm 4.2. To know how much time a content should stay in the cache, the LRU policy is considered. Let ρ is the number of contents in the cache, a content, selected for eviction, is denoted by τ_e and the last hit on the content occurred at time τ_h, then the eviction time of content from the cache is calculated as [100]:

$$E_{\rho} = \tau_e - \tau_h \tag{4.16}$$

4.3.4 Fuzzy MADM Approach for Content Eviction

The second part of our proposed strategy, i.e., Content Eviction Mechanism (CEM), is based on a method called TOPSIS [135, 206]. In TOPSIS method, the selected alternative has the longest path from the negative-ideal solution and the shortest path from the ideal solution [135]. According to our fuzzy MADM TOPSIS algorithm, the evicted content will be placed at a router having second highest outgoing interfaces, which would also be near to the subscriber. Consider the topology in Figure 4.2, router R2 has the maximum outgoing interfaces, therefore, all contents are cached here. However, if the cache of this router becomes full and a new content arrives, LRU policy will evict a content from the cache to accommodate the new arrived one. Now, the evicted content will be cached at a router which has the second highest number of interfaces. In the given figure, R3 fulfills this criteria, so will be selected for caching, which is nearer to the subscriber (if subscriber is connected to router R5 or R6).

To solve MADM problems, TOPSIS is one of the popular methods which has been used widely in several decision making problems. In our work, TOPSIS method has been modified by defining a notion of path for measuring content popularity. Because contents in ICN are fuzzy numbers, therefore, ranking of fuzzy popular contents would be a challenge which can be solved within the context of a given problem.

In our proposed strategy, for content eviction, fuzzy triangular values [206] as in [137] are used and the basic operations of fuzzy numbers for the purpose of the fuzzy MADM approach are briefly explained.

Definition Let ρ be a fuzzy triangular number defined as [135, 137]:

$$\rho = (\rho^l, \rho^m, \rho^u) \tag{4.17}$$

where $l \leq m \leq u$, and l is being the lower value and u the upper value.

The procedure for the fuzzy TOPSIS MADM approach is as [135, 206]:

Step 1: Calculate the normalized decision matrix where each normalized value r_{ij} is calculated as:

$$r_{ij} = \frac{\rho_{ij}}{\sqrt{\Sigma \rho_{ij}^2}},\ i = 1, 2, \cdots, m,\ j = 1, 2, \cdots, n. \tag{4.18}$$

where $r_{ij} = (r_{ij}^l, r_{ij}^m, r_{ij}^u)$.

Step 2: Calculate the weighted normalized decision matrix

$$d_{ij} = (\omega_j r_{ij}). \tag{4.19}$$

Step 3: Determine the best solution (A_β) and the worst solution (A_γ):

$$A_{\beta} = \{[min(d_{ij}|j\varepsilon J^{'})|i=1,2,\cdots,m],[max(d_{ij}|j\varepsilon J^{+})|i=1,2,\cdots,m]\} \equiv \{d_{\beta j}|j=1,2,\cdots,n\}, \quad (4.20)$$

$$A_{\gamma} = \{[max(d_{ij}|j\varepsilon J^{'})|i=1,2,\cdots,m],[min(d_{ij}|j\varepsilon J^{+})|i=1,2,\cdots,m]\} \equiv \{d_{\gamma j}|j=1,2,\cdots,n\}, \quad (4.21)$$

where $J^{'} = \{j = 1,2,\cdots,n\}j$ corresponding to the criteria having a negative effect, and

$J^{+} = \{j = 1,2,\cdots,n\}j$ corresponding to the criteria having a positive effect.

Step 4: Find the distance between the cached content P_i and the best condition (A_{β}) and worst condition (A_{γ}):

$$D_{i\beta} = \sqrt{\sum_{j=1}^{n}(d_{ij} - d_{\beta j})^2}, i = 1,2,\cdots,m, \quad (4.22)$$

and

$$D_{i\gamma} = \sqrt{\sum_{j=1}^{n}(d_{ij} - d_{\gamma j})^2}, i = 1,2,\cdots,m, \quad (4.23)$$

where D_{β} and D_{γ} are the distances from content P_i to the best and worst conditions, respectively.

Step 5: Find similarity to the best condition:

$$S_{i\beta} = \frac{D_{i\beta}}{(D_{i\gamma} + D_{i\beta})}, 0 < S < 1, i = 1, 2, \cdots, m. \tag{4.24}$$

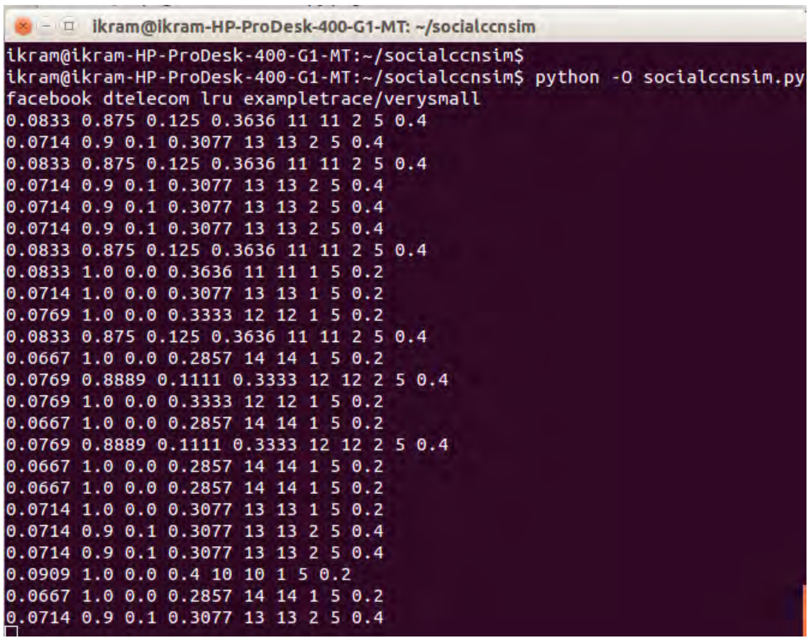
where $S_{i\beta} = 1$, in case of best condition, and

$S_{i\beta} = 0$, in case of worst condition.

Step 6: Rank the contents according to $S_{i\beta}(i = 1, 2, \cdots, m)$.

4.4 FlexPop Parameters Selection

The results of FlexPop depend upon various parameters, i.e., threshold value, comparison table (CT), Zipf distribution, cache size, catalog size, and simulation time. The threshold value is set to forward content requests from popularity table (PT) to CT. This value is set to 2 to avoid overflow as every incoming request has popularity value 1. The CT consists of those requests whose popularity value is $\geqq 2$. Based on the popularity values the contents are recommended for caching. Zipf distribution counts content probability, where skewness of the distribution is represented by α. Popular contents will receive large portion of the Internet traffic if the value of α is greater [207]. As more than half of the overall Internet traffic is attracted by multimedia applications [13] and FlexPop is designed for multimedia traffic, therefore our strategy will perform well with large α value. Cache size determines the space available in the content store for caching contents. This parameter largely affects the network performance such that if the cache size is low then maximum content eviction happens.



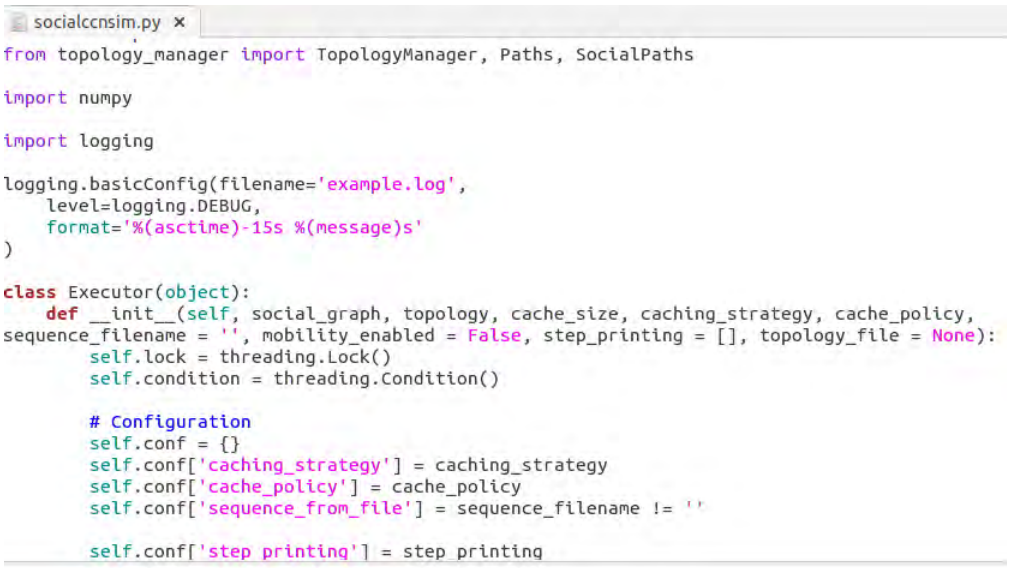
```
ikram@ikram-HP-ProDesk-400-G1-MT: ~/socialccnsim
ikram@ikram-HP-ProDesk-400-G1-MT:~/socialccnsim$
ikram@ikram-HP-ProDesk-400-G1-MT:~/socialccnsim$ python -O socialccnsim.py
facebook dtelecom lru exampletrace/verysmall
0.0833 0.875 0.125 0.3636 11 11 2 5 0.4
0.0714 0.9 0.1 0.3077 13 13 2 5 0.4
0.0833 0.875 0.125 0.3636 11 11 2 5 0.4
0.0714 0.9 0.1 0.3077 13 13 2 5 0.4
0.0714 0.9 0.1 0.3077 13 13 2 5 0.4
0.0714 0.9 0.1 0.3077 13 13 2 5 0.4
0.0833 0.875 0.125 0.3636 11 11 2 5 0.4
0.0833 1.0 0.0 0.3636 11 11 1 5 0.2
0.0714 1.0 0.0 0.3077 13 13 1 5 0.2
0.0769 1.0 0.0 0.3333 12 12 1 5 0.2
0.0833 0.875 0.125 0.3636 11 11 2 5 0.4
0.0667 1.0 0.0 0.2857 14 14 1 5 0.2
0.0769 0.8889 0.1111 0.3333 12 12 2 5 0.4
0.0769 1.0 0.0 0.3333 12 12 1 5 0.2
0.0667 1.0 0.0 0.2857 14 14 1 5 0.2
0.0769 0.8889 0.1111 0.3333 12 12 2 5 0.4
0.0667 1.0 0.0 0.2857 14 14 1 5 0.2
0.0667 1.0 0.0 0.2857 14 14 1 5 0.2
0.0714 1.0 0.0 0.3077 13 13 1 5 0.2
0.0714 0.9 0.1 0.3077 13 13 2 5 0.4
0.0714 0.9 0.1 0.3077 13 13 2 5 0.4
0.0909 1.0 0.0 0.4 10 10 1 5 0.2
0.0667 1.0 0.0 0.2857 14 14 1 5 0.2
0.0714 0.9 0.1 0.3077 13 13 2 5 0.4
```

Figure 4.5. Implementation of FlexPop in SocialCCNSim

Consequently, contents are rapidly replaced which reduces cache hit ratio and thus extra content delay is experienced. Catalog size represents the total available contents in the network. For the fair evaluation, the FlexPop strategy is examined against minimum ($1GB$) and maximum ($10GB$) cache sizes and catalog size is chosen as 10^6 contents. Simulation time demonstrates the time set for running the caching strategies in the simulator. It plays an important role for accurately simulating caching strategies. The SocialCCNSim caches the contents according to the statistical simulation of the strategy, for which the simulator needs to warm up, therefore, we have run the simulation one hour for each cache size.

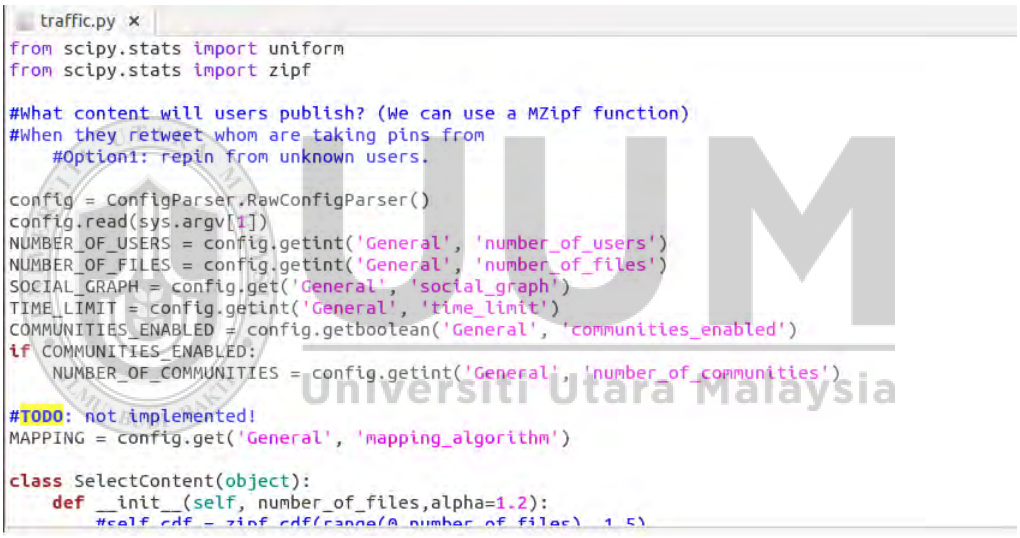
4.4.1 Verification and Validation of FlexPop

The primary concern of verification is to ensure that the proposed FlexPop strategy has been programmed correctly in SocialCCNSim simulation environment. Verification was done following the techniques presented in Chapter Three. To confirm the verification, a snapshot of the FlexPop implementation, SocialCCNSim code, SONETOR code, and SONETOR Trace file generation code are presented in Figures 4.5-



```
from topology_manager import TopologyManager, Paths, SocialPaths

import numpy

import logging

logging.basicConfig(filename='example.log',
    level=logging.DEBUG,
    format='%(asctime)-15s %(message)s'
)

class Executor(object):
    def __init__(self, social_graph, topology, cache_size, caching_strategy, cache_policy,
sequence_filename = '', mobility_enabled = False, step_printing = [], topology_file = None):
        self.lock = threading.Lock()
        self.condition = threading.Condition()

        # Configuration
        self.conf = {}
        self.conf['caching_strategy'] = caching_strategy
        self.conf['cache_policy'] = cache_policy
        self.conf['sequence_from_file'] = sequence_filename != ''

        self.conf['step_printing'] = step_printing
```

Figure 4.6. SocialCCNSim Code



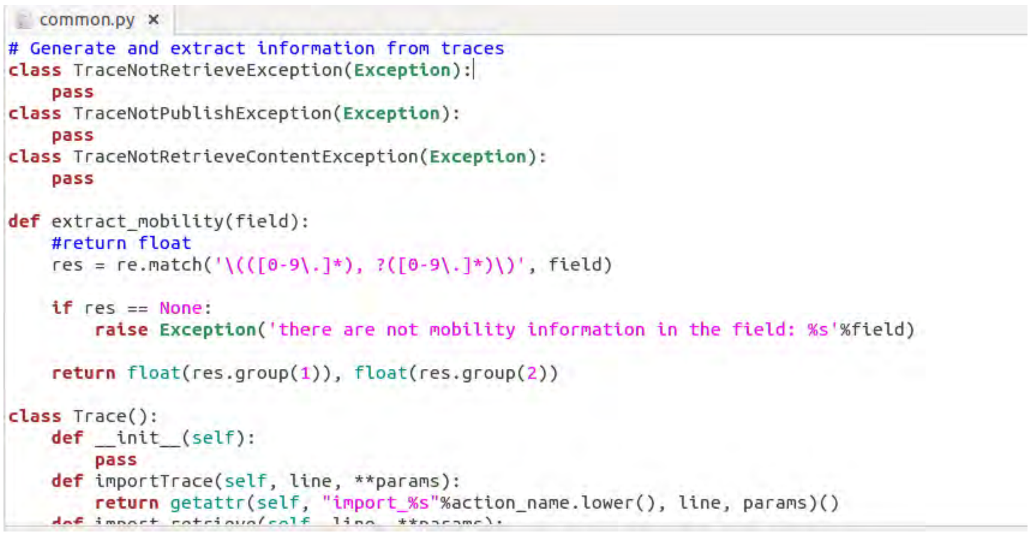
```
from scipy.stats import uniform
from scipy.stats import zipf

#What content will users publish? (We can use a MZipf function)
#When they retweet whom are taking pins from
    #Option1: repin from unknown users.

config = ConfigParser.RawConfigParser()
config.read(sys.argv[1])
NUMBER_OF_USERS = config.getint('General', 'number_of_users')
NUMBER_OF_FILES = config.getint('General', 'number_of_files')
SOCIAL_GRAPH = config.get('General', 'social_graph')
TIME_LIMIT = config.getint('General', 'time_limit')
COMMUNITIES_ENABLED = config.getboolean('General', 'communities_enabled')
if COMMUNITIES_ENABLED:
    NUMBER_OF_COMMUNITIES = config.getint('General', 'number_of_communities')

#TODO: not implemented!
MAPPING = config.get('General', 'mapping_algorithm')

class SelectContent(object):
    def __init__(self, number_of_files,alpha=1.2):
        #self.cdf = zipf.cdf(range(0,number_of_files), 1.5)
```

Figure 4.7. SONETOR Code

4.8, respectively, which confirm that:

- Both mechanisms of FlexPop, i.e., Content Placement Mechanism (CPM) and Content Eviction Mechanism (CEM) were programmed correctly, and
- The Python code, run on the Ubuntu platform, is free of bugs and errors.

Next, the FlexPop strategy was validated to ensure that it meets the intended require-



```
# Generate and extract information from traces
class TraceNotRetrieveException(Exception):
    pass
class TraceNotPublishException(Exception):
    pass
class TraceNotRetrieveContentException(Exception):
    pass

def extract_mobility(field):
    #return float
    res = re.match('\(([0-9\.]*), ?([0-9\.]*)\)', field)

    if res == None:
        raise Exception('there are not mobility information in the field: %s'%field)

    return float(res.group(1)), float(res.group(2))

class Trace():
    def __init__(self):
        pass
    def importTrace(self, line, **params):
        return getattr(self, "import_%s"%action_name.lower(), line, params)()
```

Figure 4.8. SONETOR Trace File Code

ments in terms of cache hit rate, memory utilization, redundancy, content retrieval delay, and stretch. For surety, the FlexPop strategy was run on four different topologies, i.e., Abilene, Tiger, GEANT, and DTelekom; and the obtained results were compared with the three popular strategies, named Leave Copy Everywhere (LCE), Probabilistic Caching (ProbCache), and Most Popular Content (MPC). The validation was done using SocialCCNSim and the value of Zipf probability model α parameter was set as 2.0. The simulations were run for 10 times on each topology and the average value of cache hit rate was obtained for LCE, ProbCache, MPC, and FlexPop strategies. The

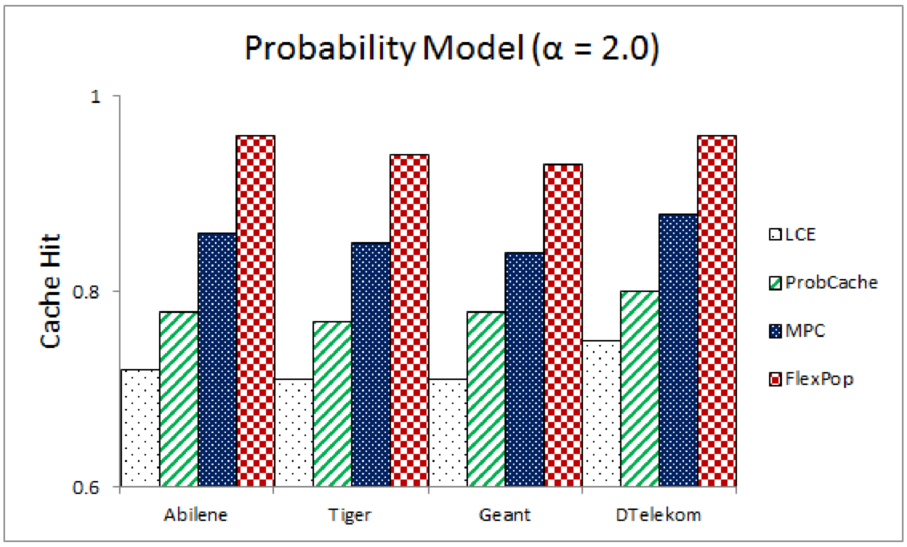


Figure 4.9. Cache Hit Rate on Different Topologies

Table 4.1
Cache Hit: Analysis vs Simulation with $\alpha = 0.7$ and Chunk Size 10MB

Topology	**Cache Size (GB)**	**No. of Chunks (MB)**	**Analysis**	**Simulation**	**Difference**
Facebook	1	$1(10^2)$	0.10	0.10	0.0
	2	$2(10^2)$	0.12	0.11	0.01
	3	$3(10^2)$	0.14	0.12	0.02
	4	$4(10^2)$	0.15	0.13	0.02
	5	$5(10^2)$	0.16	0.14	0.02
	6	$6(10^2)$	0.17	0.14	0.03
	7	$7(10^2)$	0.18	0.15	0.03
	8	$8(10^2)$	0.19	0.15	0.04
	9	$9(10^2)$	0.19	0.16	0.03
	10	$10(10^2)$	0.20	0.16	0.04

obtained resultant graph is presented in Figure 4.9.

4.4.2 Simulation vs. Analysis

To know the accuracy of system model and simulation, the analysis is done in Maple 18 for cache hit according to the parameters presented in Table 4.1 and 4.2. For the analysis of cache hit, we consider the real Facebook topology [165, 166, 167] which consists of 4,039 nodes. We assume that each node in the network is placed at a constant distance: in our assumption this distance is 25 meters. Each time when a content is downloaded, the hop decrement increases 100 hops. This assumption is made on the basis of our ordinary topology in Figure 4.2, where initially a subscriber is 4 hops away (on any path) from the node having the desired content (i.e., router R1). As the hit occurs, the content is cached on the node having the highest number of outgoing interfaces (Router R2 in the given figure), which is 3 hops away from the user(s).

When $0 < \alpha < 1$, the asymptotic cache hit ratio H_c is calculated as [208, 209]:

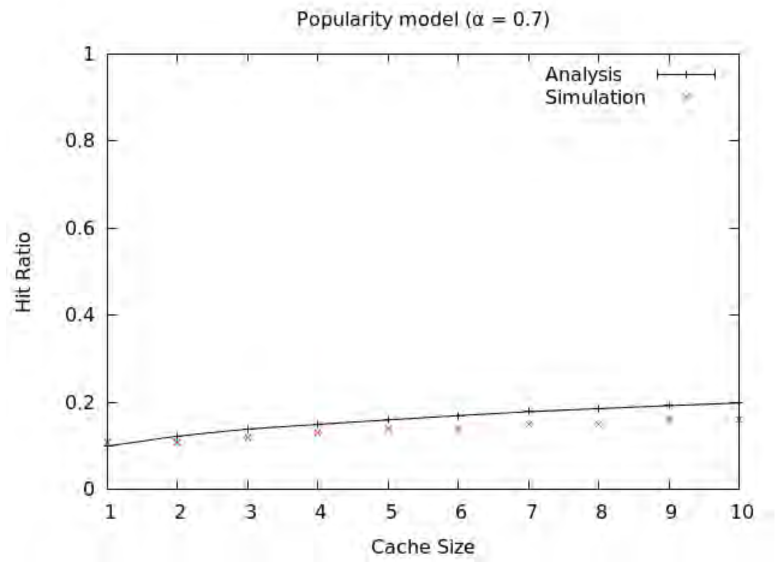


Figure 4.10. Cache Hit: Analysis vs Simulation with $\alpha = 0.7$

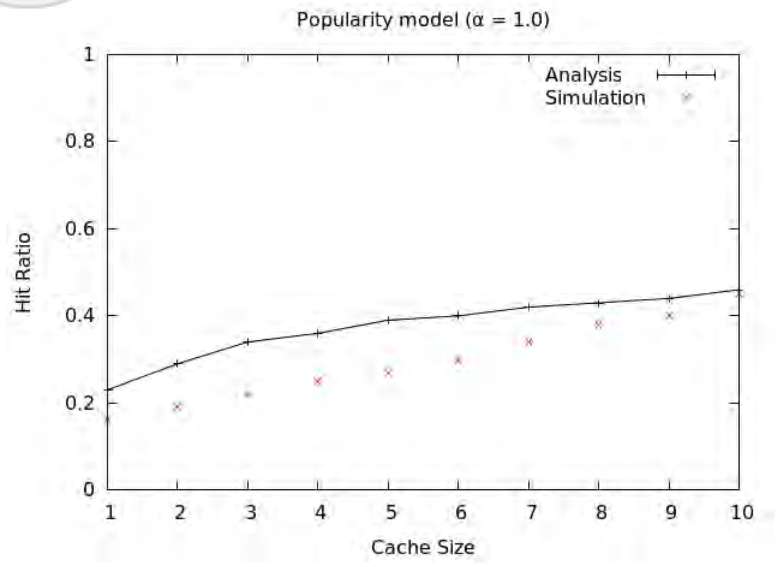


Figure 4.11. Cache Hit: Analysis vs Simulation with $\alpha = 1.0$

Table 4.2
Cache Hit: Analysis vs Simulation with $\alpha = 1.0$ and Chunk Size 10MB

Topology	**Cache Size (GB)**	**No. of Chunks (MB)**	**Analysis**	**Simulation**	**Difference**
Facebook	1	$1(10^2)$	0.23	0.16	0.07
	2	$2(10^2)$	0.29	0.19	0.10
	3	$3(10^2)$	0.34	0.22	0.12
	4	$4(10^2)$	0.36	0.25	0.11
	5	$5(10^2)$	0.39	0.27	0.12
	6	$6(10^2)$	0.40	0.30	0.10
	7	$7(10^2)$	0.42	0.34	0.08
	8	$8(10^2)$	0.43	0.38	0.05
	9	$9(10^2)$	0.44	0.40	0.04
	10	$10(10^2)$	0.46	0.45	0.01

$$H_c = C^{1-\alpha} \tag{4.25}$$

where C is the cache size and $\alpha = 0.7$. Looking at the analysis and simulation results, presented in Table 4.1, the average result of analysis is 16% while it is 13.6% for simulation. The average difference is 2.4% and hence the accuracy is 97.6%. The resultant graph is shown in Figure 4.10. The same variables, i.e., cache size and chunk size, are used for the scenario when $\alpha = 1.0$. Now, if $\alpha = 1.0$, the asymptotic cache hit ratio H_c is calculated as [208, 209]:

$$H_c = lnC. \tag{4.26}$$

The analysis and simulation result is presented in Table 4.2 and Figure 4.11. The average analysis and simulation results are 37.6% and 29.6%, respectfully. The average difference is 8% and therefore the accuracy is 92%. It is observed that when the cache size is small the difference of analysis and simulation results (with $\alpha = 1.0$) is high, however, with the increase of cache size this difference approaches 0.

Similarly, for the analysis of stretch results, we consider the same scenario with real Facebook topology [165, 166, 167]. The numerical results can be calculated as [210]:

$$S = \frac{\alpha n}{2\gamma} \tag{4.27}$$

where S represents Stretch, n is the total number of network nodes, γ is the distance between two hops, and α is the Zipf probability parameter, such that $0 < \alpha < 1$.

In addition, the numerical results for $\alpha = 1.0$ can be calculated as:

$$S = \frac{\alpha n}{2\gamma} - (D+h) \tag{4.28}$$

where n is the number of total network nodes, γ is the distance between two hops, D is the hop decrement, and h is constant. We assume that $D = \gamma$ and the value of $h = 4$. This assumption is based on our ordinary proposed topology, shown in Figure 4.2. The achieved analysis results are presented in Table 4.3 and Figure 4.12 for $\alpha = 0.7$, and Table 4.4 and Figure 4.13 for $\alpha = 1.0$. The average difference in $\alpha = 0.7$ is 2.2% while it is 4.8% when $\alpha = 1.0$.

Figure 4.14 illustrates the ratio of *Cache Hit* in FlexPop using different values of popularity model, α. In the given figure x-axis shows the cache size ranging from 1GB to 10GB, while y-axis is the average hit ratio of cached content items. Certain increase

Table 4.3
Stretch: Analysis vs Simulation with $\alpha = 0.7$

Topology	Cache Size (GB)	Analysis	Simulation	Difference
Facebook	1	0.56	0.58	0.02
	2	0.51	0.54	0.03
	3	0.50	0.53	0.03
	4	0.49	0.52	0.03
	5	0.48	0.50	0.02
	6	0.47	0.49	0.02
	7	0.46	0.48	0.02
	8	0.45	0.47	0.02
	9	0.44	0.46	0.02
	10	0.43	0.44	0.01

Table 4.4
Stretch: Analysis vs Simulation with $\alpha = 1.0$

Topology	Cache Size (GB)	Analysis	Simulation	Difference
Facebook	1	0.51	0.54	0.03
	2	0.47	0.52	0.05
	3	0.43	0.48	0.05
	4	0.39	0.44	0.05
	5	0.35	0.40	0.05
	6	0.31	0.36	0.05
	7	0.27	0.32	0.05
	8	0.23	0.28	0.05
	9	0.19	0.24	0.05
	10	0.15	0.20	0.05

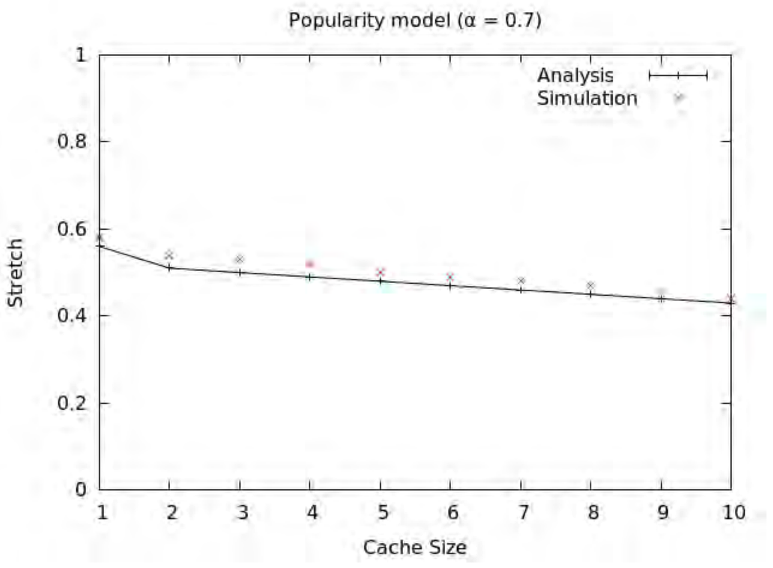


Figure 4.12. Stretch: Analysis vs Simulation with $\alpha = 0.7$

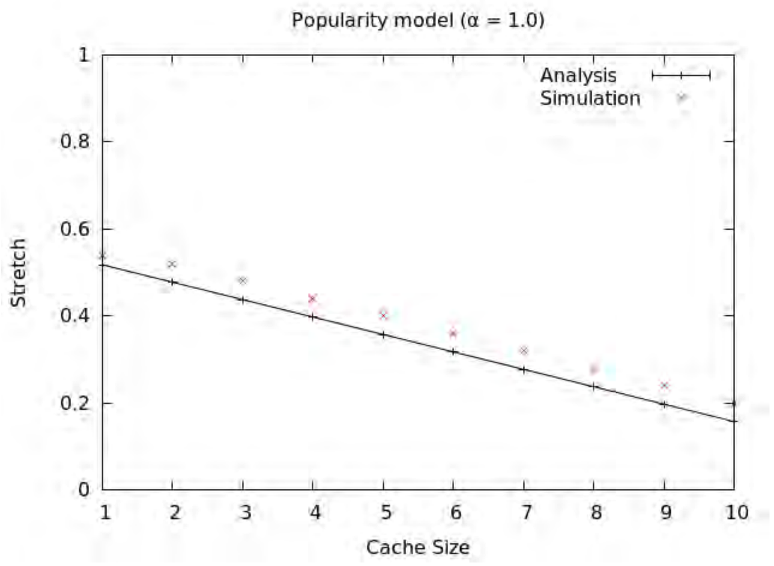


Figure 4.13. Stretch: Analysis vs Simulation with $\alpha = 1.0$

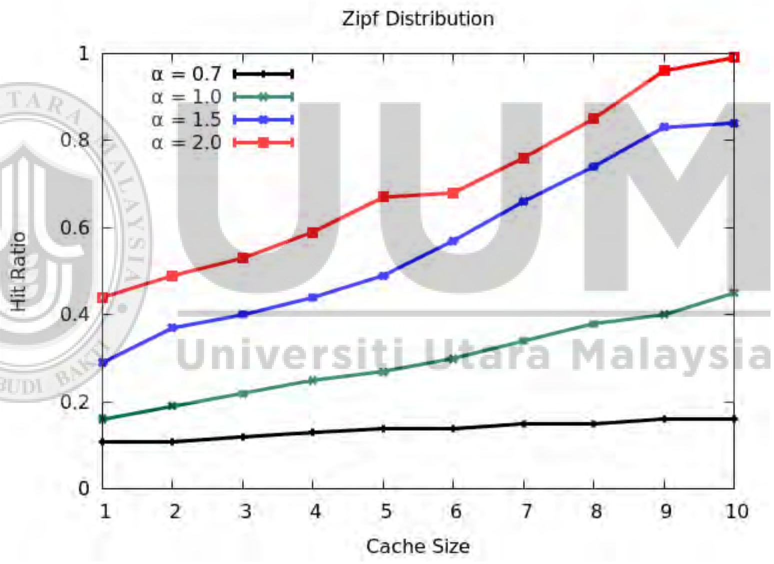


Figure 4.14. Probability of Cache Hit by Zipf Popularity Model α

can be found in cache hit ratio with the increase of popularity value α from 0.7 to 2.0 and cache size from 1GB to 10GB, respectively.

4.5 Summary

This chapter introduced a caching strategy for the future Internet, named Flexible Popularity-based Caching (FlexPoP). The chapter also presented the basic idea of content placement and eviction, and the need for a flexible caching strategy, which can better utilize memory and increase cache hit rate as well as reduce content retrieval delay, stretch, and redundancy.

If a new content arrives and the cache of a content store is full then it needs to replace one of the stored contents. Since Content Placement Mechanism (CPM) cannot solve this issue, therefore, a Content Eviction Mechanism (CEM) is required to handle this problem.

CHAPTER FIVE
SIMULATION RESULTS

In Chapter Four, the FlexPop strategy was introduced, which consists of two mechanisms, i.e., CPM and CEM, for content placement and eviction in ICN. Moreover, the chapter discussed two access methods, i.e., SAW, which is implemented for popular content caching, and TOPSIS that was used for content eviction. This chapter presents the performance evaluation of FlexPop strategy in full detail as well as its comparison with LCE, ProbCache, and MPC caching strategies. The chapter starts with introducing the simulation environments for ICN caching in Section 5.1. Outcomes of the performance evaluation of FlexPop strategy in comparison with those of LCE, ProbCache, and MPC are discussed in Section 5.2. Topological affect on the simulation results is described in Section 5.3 and the general discussion on simulation results related to the technical factors is outlined in Section 5.4. Finally, the chapter is concluded with the summary presented in Section 5.5.

5.1 Simulation Environments for Caching Strategies in ICN

Since the day when the concept of ICN was proposed in 2009 [30], it became a very popular research topic with a bunch of new proposals presented in renowned conferences and published in reputed journals. A little number of these proposals deals with routing schemes [211], congestion control [212], and content namespace [213], but majority of them focus on the in-network caching strategies as caching is one of the major and critical issues in the ICN architecture.

With respect to caching strategies, usually researchers evaluate their design policies according to their simulation scenarios and choose their configuration parameters at their own choice, which make it unfeasible to compare the difference among caching strategies. Although a lot of simulation scenarios have been proposed for evaluating ICN caching strategies, in this section we consider four main simulation parameters as in [174, 58], i.e., network topology, content popularity model, cache size, and catalog size.

5.1.1 Network Topology

In the literature, presented in Chapter Two, the performance of ICN caching strategies have been tested using different topologies. Saino et al. [96] consider four topologies, i.e., GEANT (European academic network), WIDE (Japanese academic network), Tiscali (pan-European commercial ISP), and GARR (Italian academic network); Psaras et al. [41] use a 6-level binary tree comprising 127 nodes, where the last two levels of ISP issue all the requests and the root node serves these requests; Li et al. [102] use a 4-level tree topology, where the total size of content stores of each router ranges from 10% to 50% of the whole contents; Chai et al. [106] use a 4 to 6-levels k-ary tree and each node has either 2 or 5 children; Rossi et al. [174] use five topologies, i.e., Abilene, Level3, Tiger, GEANT, and DTelekom; Cho et al. [110] use ISP-level topology that includes 1 transit and 5 stub domains. The transit domain consists of 5 routers while each stub domain has 10 routers. The topology also has 10 servers which serve requests from 1,000 connected users; Bernardini et al. [33] use four ISP-level topologies, i.e., GEANT, Tiger, Abilene, and DTelekom.

For the fair evaluation of caching strategies, there is no general agreement on the topology selection. It is therefore a critical issue because topology has a direct impact on the simulation results [174]. Abilene, GEANT, Tiger, and DTelekom are ISP-level topologies, where Abilene consists of 11 nodes, GEANT and Tiger 22 nodes each, and DTelekom 68 nodes, as presented in Figure 5.1.

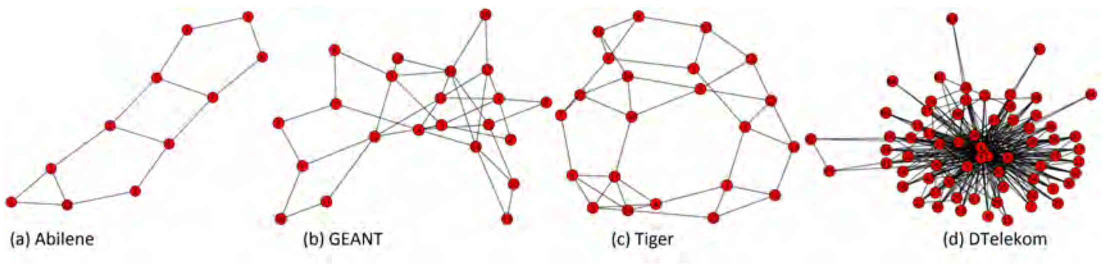


Figure 5.1. ISP-level Topologies [58]

5.1.2 Content Popularity Model

The content popularity model refers to a utility that records the popularity value of every individual content. In other words, it is a function which counts the number of requests for every single content. The previous conducted research shows that out of millions only a few websites attracted the majority of customers [214, 215]. Therefore, the content popularity can be modeled with a probability distribution function as a power law [58]. In most of the simulations and models, the commonly used distributions are Zipf and its generalized form Maldelbrot-Zipf (MZipf), which belong to the power law distribution [216, 174, 58]:

$$pmf(x,N,q,\alpha) = \frac{(q+x)^{-\alpha}}{\sum_{k=1}^{N}(q+k)^{-\alpha}} \tag{5.1}$$

where α is slope of the distribution, q denotes its plateau, N is the number of elements, and pmf represents the probability mass function. When q tends to zero as in [58], a flatter distribution is achieved. In the literature, the value of α parameter for Zipf or MZipf ranges from 0.5 to 3.0. In simulation based on MZipf distribution, a content is selected with its probability value when it is requested. Some contents may be requested more times than others based on configuration of the α parameter.

We discovered a variety of values for the α parameter in the literature. Saino et al.

[96] use a Zipf popularity model with the values of α varies between 0.6 and 1.1. In [41], the value of this parameter is 0.8. Li et al. [102] and Zhang et al. [103] evaluate with 0.3 and 0.9 for α. In [106] the value of this parameter is 1.0. Sung et al. [107] evaluate with an α between 0.25 and 1.5, while in [109] it is 0.6 and 1.2. In [110], the α is evaluated for 0.85. Bernardini et al. [33, 58, 217] use the values between 0.65 and 2.0, while Ong et al. [114] take this value between 0.5 and 3.0. In [108], the α is 0.7, while it varies between 0.7 and 0.96 in [122]. In [174], α ranges between 0.6 and 2.5. Other researchers, for example, Fayazbakhsh et al. [119] evaluate the caches with traces extracted from the Akamai CDN Asia [58]. To model content popularity, MZipf distribution is considered the best nominee and therefore no agreement is made on the value of its α parameter. As we discovered that this value ranges from 0.5 to 3.0, it may have a great effect on the performance evaluation of the caching strategies.

5.1.3 Cache Size

The cache size specifies the available space in every node for temporally storing content objects. Cache size is either denoted with absolute value or a ratio with respect to the catalog size, i.e., 20 elements or 0.02 of the catalog size [58]. Saino et al. [96] use cache size 0.2. Cho et al. in [110] and Chai et al. in [106] use cache size 0.1. Rosensweig et al. [101] use the cache size from 0.1 to 0.4, while Li et al. [102] take this value from 0.1 to 0.5. Zhang et al. [103] take this value 100 while Bilal and Kang [109] consider cache size from 100 to 1,000 elements. Sung et al. [107] use 2,500 elements, each of size 2GB, and the cache space is set to 1TB. Wang et al. in [108] simulate the scenarios using the cache capacity of 50Mb and 200Mb with the content size of 100Mb and 300Mb. Bernardini et al. [33, 58] evaluate their experiments with cache size ranges between 1 and 1,000 elements correspond to 10^{-6} to 10^{-3} as cache size ratio, while Ong et al. [114] use cache size 0.05 to 0.3.

Table 5.1
Simulation Scenario

Parameter	**Description**
Cache Size	100-1,000 elements (1GB-10GB)
Catalog Size	10^6 elements
α Parameter	0.7, 1.0, 1.5, 2.0
Topology	Abilene
Social Network Topology	Facebook [165, 166, 167]
Replacement Policy	LRU
Simulator	SocialCCNSim [142, 173], OPNET Modeler 18.0 [143].
Simulation Time	One day

5.1.4 Catalog Size

The catalog represents the total number of contents in a network. In the existing work, different number of values have been used for the catalog size. Chai et al. [106] and Zhang et al. [103] use this value 10^3. Rossi et al. [174] consider a YouTube-like catalog, which includes 10^8 objects with average 10MB chunk. Bilal and Kang [109] take the catalog size as 10^4. Bernardini et al. [33, 58] consider in their experiments a catalog of 10^6 elements, while Ong et al. [114] use this value as 10^4. Some researchers have not given detail about the catalog size, but in [96, 41, 119, 104] the authors have only mentioned that for all catalog they generate 10^5 requests.

5.1.5 Simulation Setup

In this study we compare our results with those of the LCE, ProbCache, and MPC, therefore, to confirm that our simulation results are valid, the same simulation setup is used as in [33] to achieve the same results for the mentioned two strategies as produced in [58]. The reason for comparison with LCE, ProbCache, and MPC is such that LCE is the default ICN caching strategy, ProbCache has outperformed LCE [41], and MPC has shown superiority over most of the existing strategies, as submitted in [58]. Here, we use the Abilene topology having 8 requesting nodes with Facebook [165, 166, 167], which consists of 4,039 users, 88,234 friends, and each user counts 44 relationships. We evaluate our experiments with cache size ranges between 100 and 1,000 elements (1GB to 10GB respectively), a catalog size of 10^6 elements, and Zipf popularity model with the values of α vary between 0.7 and 2.0 (see Table 5.1). In the literature, 10^4 contents is the widely used value for catalog size [109, 114, 163, 111], but due to the nature of existing traffic it is hard to imagine Internet merely with 10^4 contents. Therefore, in our experiments the catalog size is considered as 10^6 contents, which is believed as an adequate size for measuring performance of the Internet traffic [33, 217]. The contents and elements represent files, therefore they are used interchangeably throughout this thesis. In the simulation results, the average values are considered for *Cache Hit, Content Delay, Redundancy, Memory Utilization* and *Stretch.*

5.2 Simulation Results

In this section we present our simulated work in line with the following objectives: cache hit, redundancy, content retrieval delay, memory utilization, and stretch.

5.2.1 Cache Hit

Cache Hit is the most significant metric in evaluating the performance of ICN and is represented by the following equation [58]:

$$H_c = \frac{\Sigma_{i=1}^{n} hit_i}{\Sigma_{i=1}^{n} (hit_i + miss_i)} \tag{5.2}$$

where *n* is the number of all network nodes and *hit* operation occurs if the cache of a node is traversed and the requested content is found; the *miss* refers to the situation when a node is searched for a requested content and it is not available in the cache; H_c

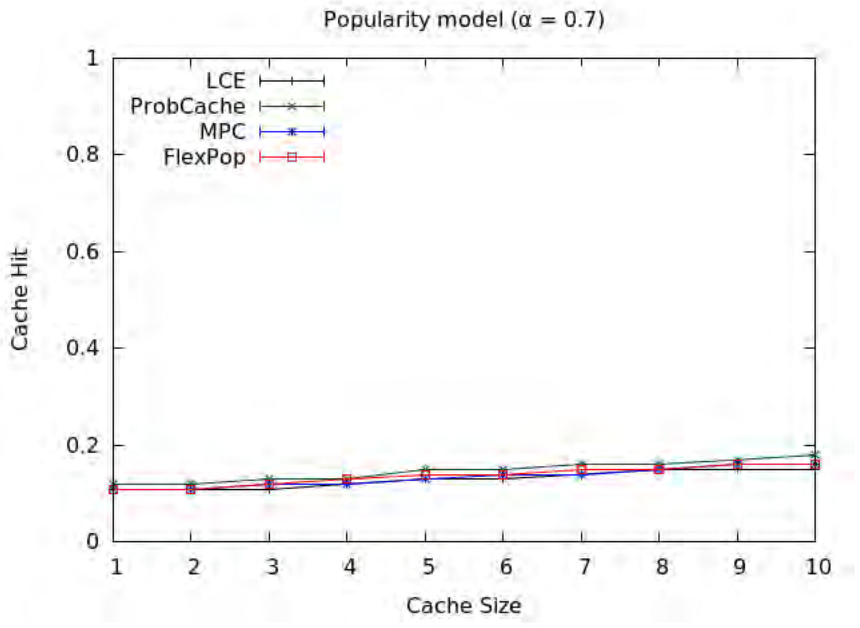


Figure 5.2. Cache Hit with $\alpha = 0.7$

is the total cache hit rate.

The ratio of cache hit is shown in Figures 5.2 to 5.5. In low popularity scenario, i.e., when $\alpha = 0.7$, the ratio of cache hit is low on cache sizes 100 to 1,000 elements (see Figure 5.2), but with the increase of probability parameter α from 0.7 to 1.0, 1.5, and 2.0 as well as the increase of cache size (Figures 5.3, 5.4, and 5.5, respectively), the higher hit ratio is achieved. In Figure 5.2, the hit ratio reached only 16% with popularity value $\alpha = 0.7$ (which is almost the same for all four strategies), however, with the increase of this value it reached 45% with $\alpha = 1.0$ (in Figure 5.3) compared to 38% of MPC, 32% of ProbCache, and 30% of LCE; 84% with $\alpha = 1.5$ (in Figure 5.4) compared to 74% of MPC, 65% of ProbCache, and 53% of LCE; and 99% with $\alpha = 2.0$ (in Figure 5.5) in comparison with 91% of MPC, 80% of ProbCache, and 78% of LCE, respectively. The numerical result is shown in Table 5.2. For short time simulations and low cache size, LCE achieved the highest hit rate, but as the cache size increased (with $\alpha = 1.5$, and 2.0), FlexPop achieved the highest hit ratio in comparison to MPC, ProbCache, and LCE.

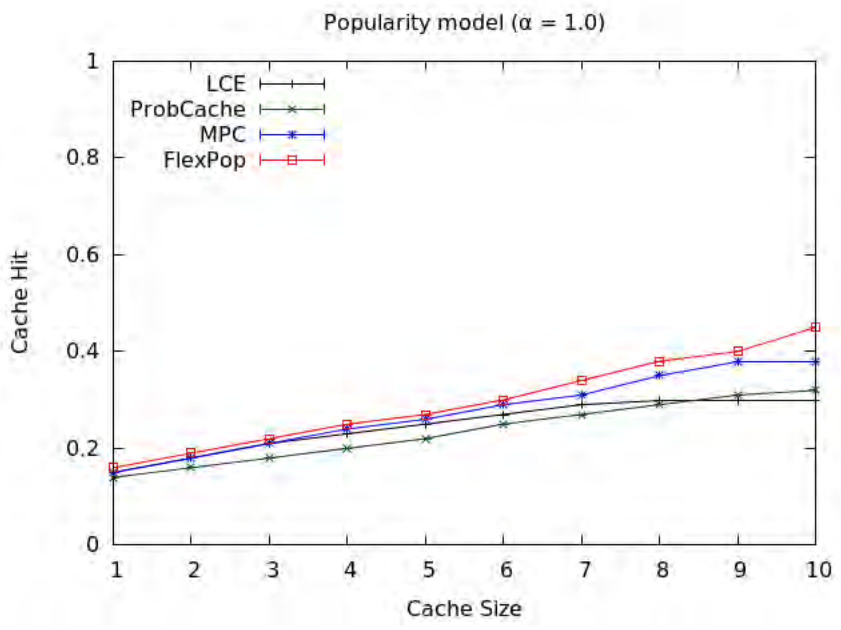


Figure 5.3. Cache Hit with $\alpha = 1.0$

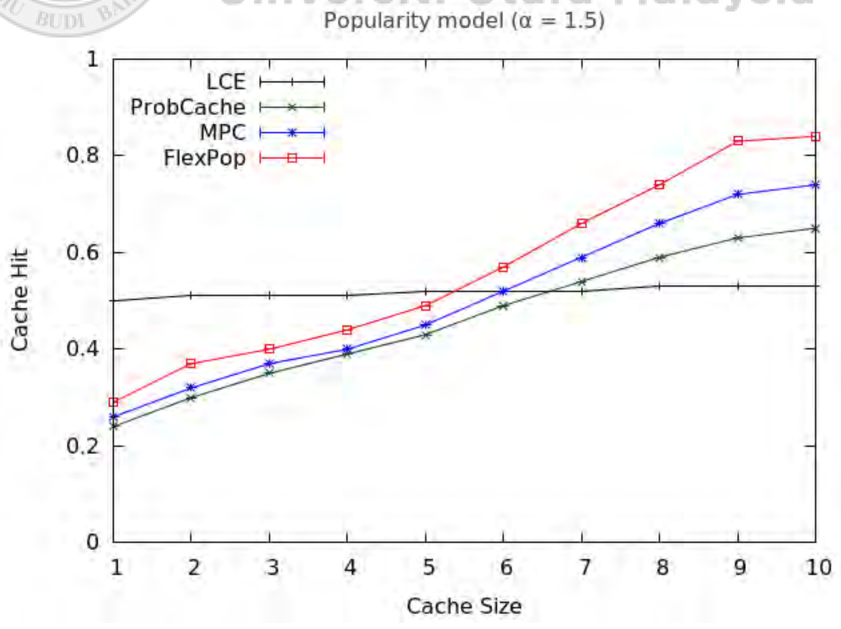


Figure 5.4. Cache Hit with $\alpha = 1.5$

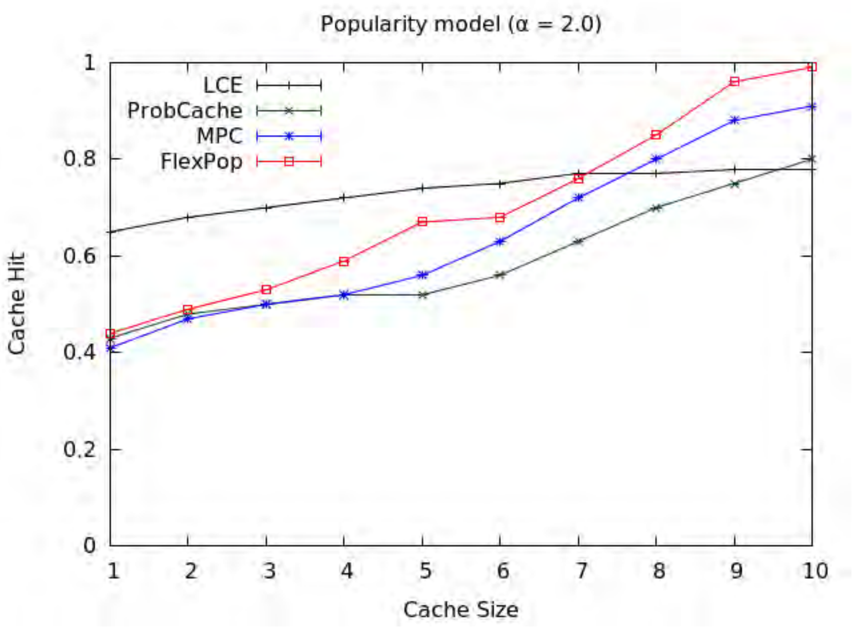


Figure 5.5. Cache Hit with $\alpha = 2.0$

Table 5.2
Cache Hit Comparison

Popularity model α				**Cache Size**		**Cache Hit**			
0.7	**1.0**	**1.5**	**2.0**	**100**	**1,000**	**LCE**	**ProbCache**	**MPC**	**FlexPop**
√	-	-	-	√	-	0.1	0.12	0.1	0.1
√	-	-	-	-	√	0.15	0.18	0.16	0.16
-	√	-	-	√	-	0.15	0.14	0.15	0.16
-	√	-	-	-	√	0.30	0.32	0.38	0.45
-	-	√	-	√	-	0.50	0.24	0.26	0.29
-	-	√	-	-	√	0.53	0.65	0.74	0.84
-	-	-	√	√	-	0.65	0.43	0.41	0.44
-	-	-	√	-	√	0.78	0.80	0.91	0.99

5.2.2 Path Redundancy

Redundancy is another salient metric to measure the performance of ICN and is shown through the following equation [218]:

$$R_{ci} = \mu_i^2 * \frac{1}{m_c} * S_i \tag{5.3}$$

and

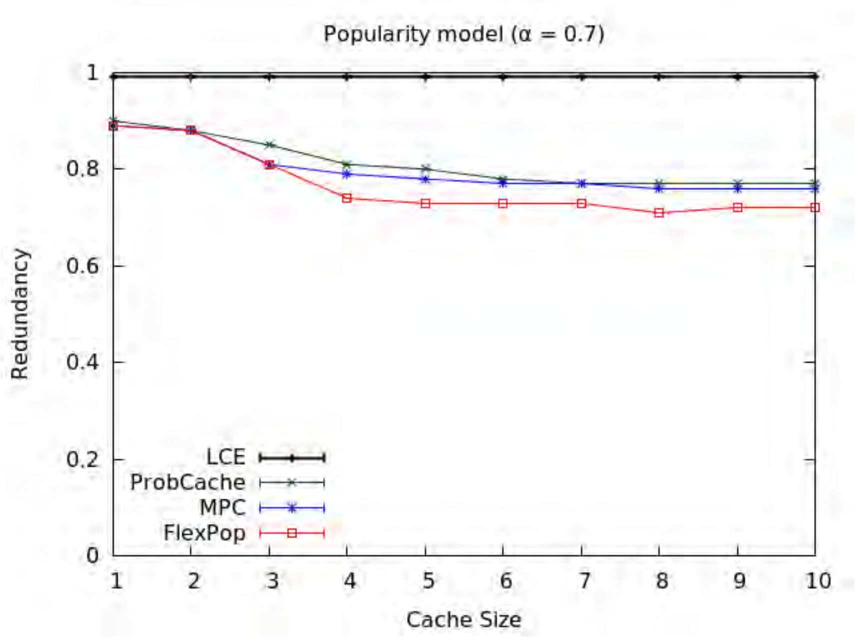


Figure 5.6. Redundancy with $\alpha = 0.7$

$$R_c = \sum_i R_{ci} \tag{5.4}$$

where R_{ci} is content redundancy given the i$^{\text{th}}$ item of the cached contents and R_c is the total redundancy. The symbol m_c stands for the total number of content items; μ_i is the i$^{\text{th}}$ content item, while S_i is the number of all stored contents on their i$^{\text{th}}$ node position.

Figures 5.6 to 5.9 show the average percentage redundancy of the cached contents in all network nodes on the path from publisher to subscriber. In low popularity scenario ($\alpha = 0.7$) the percentage of (on-path) redundant contents was 100% in LCE on all cache sizes (i.e., 100 to 1,000 elements). In MPC, this percentage was reduced from 89% (with cache size 100) to 76% (with cache size 1,000), which is almost the same as ProbCache; while in FlexPop the percentage was a little lower than MPC and ProbCache, i.e., 89% with cache size 100 to 72% with cache size 1,000 elements (see Figure 5.6). As the value of α parameter is increased from 0.7 to 1.0, 1.5, and 2.0 in Figures 5.7, 5.8, and 5.9, respectively, the percentage of redundant contents is de-

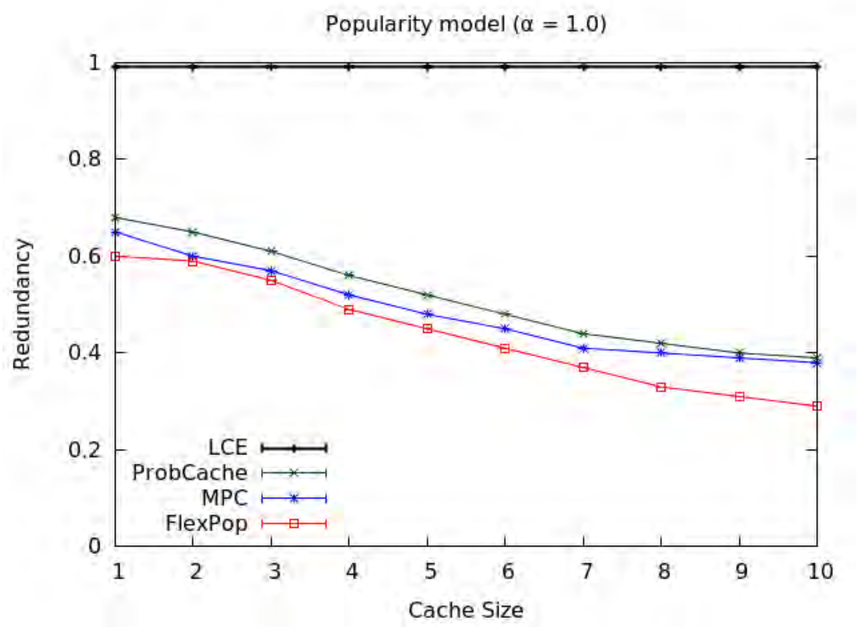


Figure 5.7. Redundancy with $\alpha = 1.0$

creased dramatically in FlexPop. That is, in LCE, this value is constant (100%) with all cache sizes (because LCE caches a copy of the content in all network nodes on the path from publisher to subscriber); in MPC this value is reduced from 65% (with cache size 100) to 38% (with cache size 1,000); in ProbCache the values were recorded as 68% and 39% with cache size 100 and 1,000, respectively; and in FlexPop, it is reduced from 60% (with cache size 100) to 29% (with cache size 1,000), as shown in Figure 5.7. With popularity value 1.5 and cache size 100 to 1,000, the redundancy is reduced from 55% to 37% in MPC, which is again almost the same as ProbCache, and 45% to 29% in FlexPop, as presented in Figure 5.8. In Figure 5.9 with cache size 100 to 1,000 and high popularity scenario (i.e., $\alpha = 2.0$), the percentage value in LCE is again constant (100%) for all cache sizes; in MPC it is reduced from 40% to 18%; in ProbCache it is reduced from 43% to 22%; while in FlexPop the average percentage was recorded with reduction from 30% to 5%. The numerical results are presented in Table 5.3.

Thus, it confirms that using FlexPop with high popularity and higher cache size, the

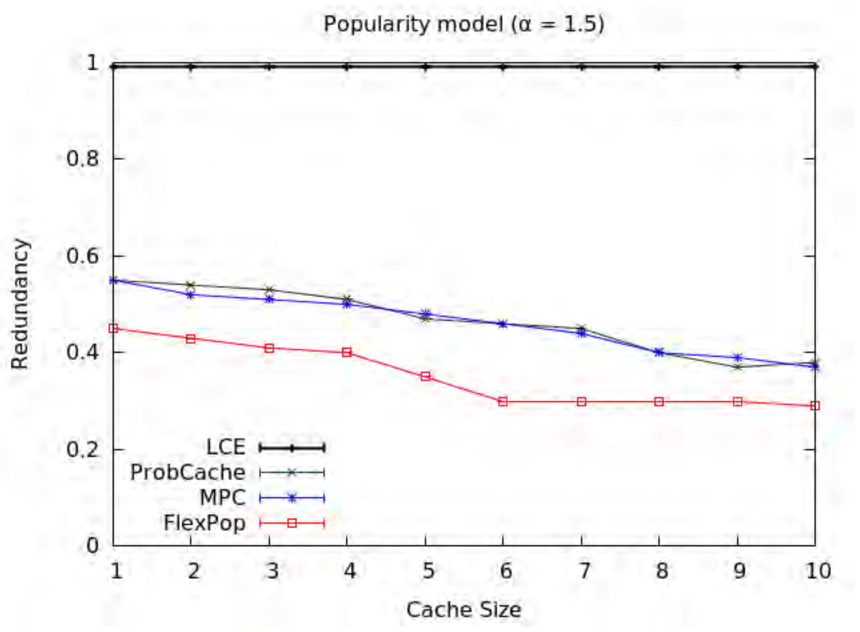


Figure 5.8. Redundancy with $\alpha = 1.5$

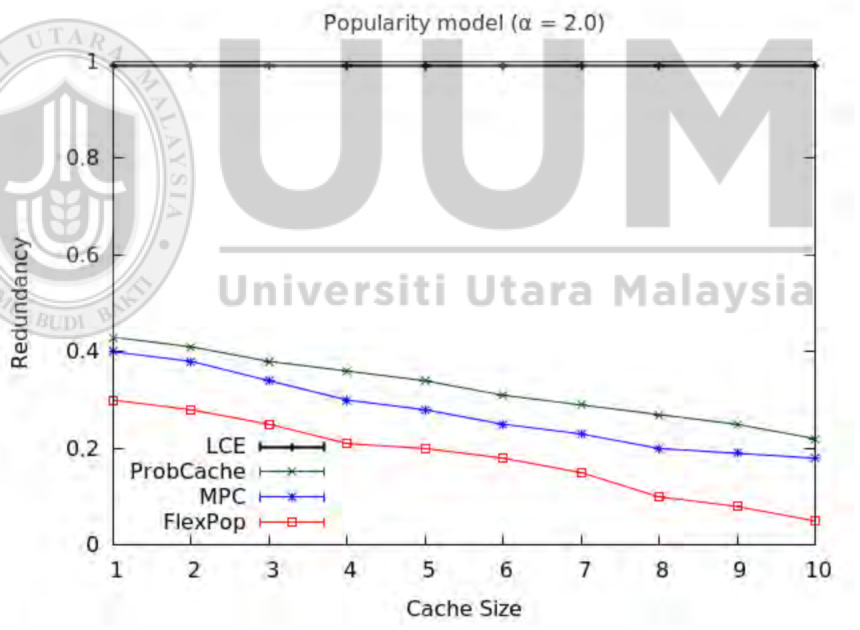


Figure 5.9. Redundancy with $\alpha = 2.0$

redundancy of cached contents can be reduced up to some extent.

5.2.3 Content Retrieval Delay

Content retrieval delay is the third crucial metric in the performance measurement of ICN which is represented by the following formula [58]:

Table 5.3
Path Redundancy Comparison

Popularity model α				**Cache Size**		**Path Redundancy**			
0.7	**1.0**	**1.5**	**2.0**	**100**	**1,000**	**LCE**	**ProbCache**	**MPC**	**FlexPop**
√	-	-	-	√	-	1.0	0.90	0.89	0.89
√	-	-	-	-	√	1.0	0.77	0.76	0.72
-	√	-	-	√	-	1.0	0.68	0.65	0.60
-	√	-	-	-	√	1.0	0.39	0.38	0.29
-	-	√	-	√	-	1.0	0.55	0.55	0.45
-	-	√	-	-	√	1.0	0.38	0.37	0.29
-	-	-	√	√	-	1.0	0.43	0.40	0.30
-	-	-	√	-	√	1.0	0.22	0.18	0.05

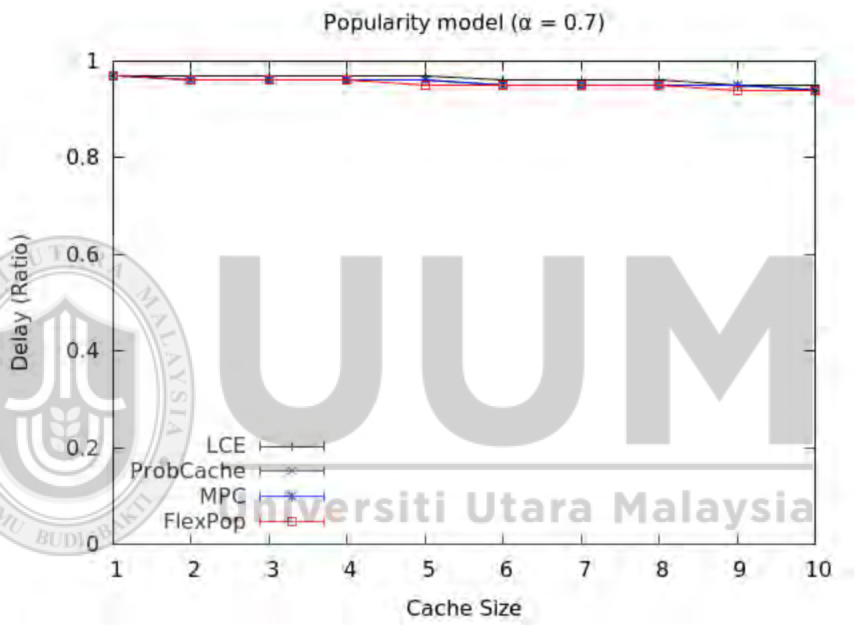


Figure 5.10. Content Retrieval Delay with $\alpha = 0.7$

$$D_c = \frac{\Sigma_{i=1}^{|R|} delayR_i}{|R|} \tag{5.5}$$

where c represents content and $|R|$ is the number of requests made by customers; D_c is the overall content delay.

Figures 5.10 to 5.13 show the ratio of average content retrieval delay in the network. In the scenarios where the content popularity is low, i.e., when the α parameter of the

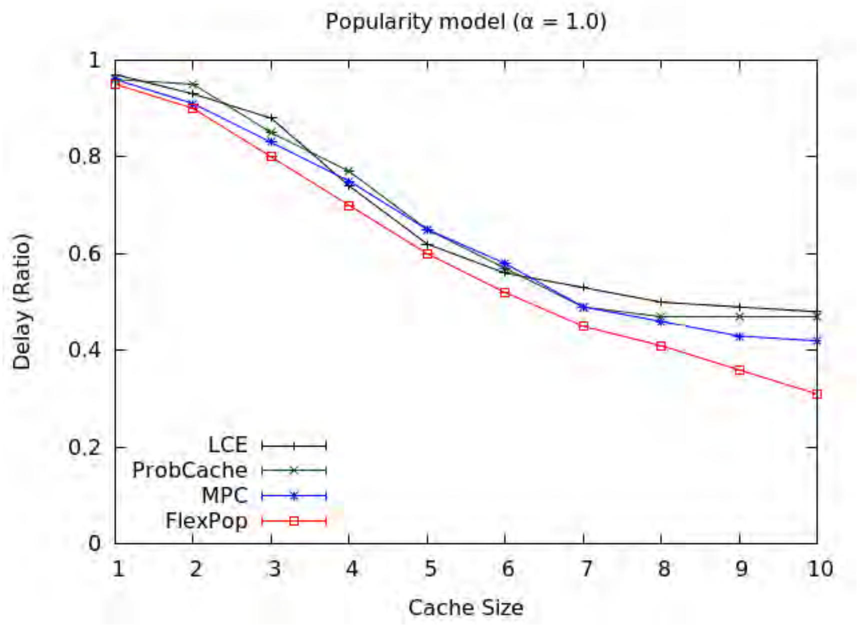


Figure 5.11. Content Retrieval Delay with $\alpha = 1.0$

Zipf probability is 0.7, the ratio of content retrieval delay is almost the same in all strategies with all cache sizes, which is somehow between 94% and 97%. However, with the increase of this parameter (from 0.7 to 1.0), the ratio of content delay was observed such that it was reduced from 97% to 48% in LCE with the increase of cache size from 100 to 1,000 elements. This percentage was reduced from 96% to 47% in ProbCache and 96% to 42% in MPC (with cache size 100 to 1,000); whereas in FlexPop, with the same cache sizes, this ratio was minimized from 95% to 31%. Similarly, with the increase of α parameter from 1.0 to 1.5 and 2.0, the content delay was recorded such as it was reduced from 72% (with cache size100 and $\alpha = 1.5$) to 30% (with cache size 1,000 and $\alpha = 1.5$) in LCE; in ProbCache this reduction was observed from 72% to 24% and in MPC it was recorded as 68% to 18% with cache size 100 and 1,000, respectively; while in FlexPop (with cache size 100), this ratio was 58%, but it was reduced to 10% with cache size 1,000. Moreover, with $\alpha = 2.0$ and cache size 100, the recorded content delay was 59% in LCE, 55% in ProbCache, 45% in MPC, and 45% in FlexPop, while (with the increase of cache size from 100 to 1,000) the ratio was recorded as follows: LCE = 20%, ProbCache = 15%, MPC =

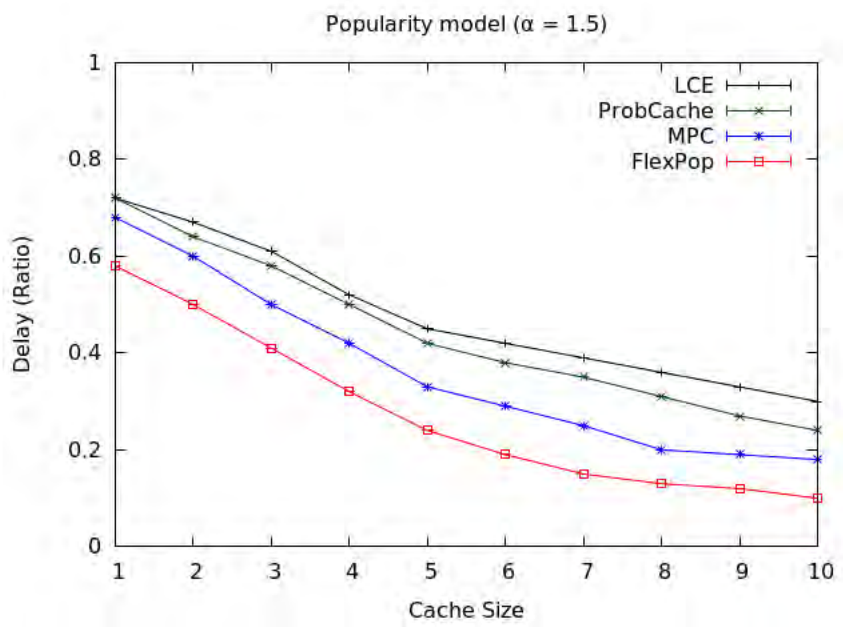


Figure 5.12. Content Retrieval Delay with $\alpha = 1.5$

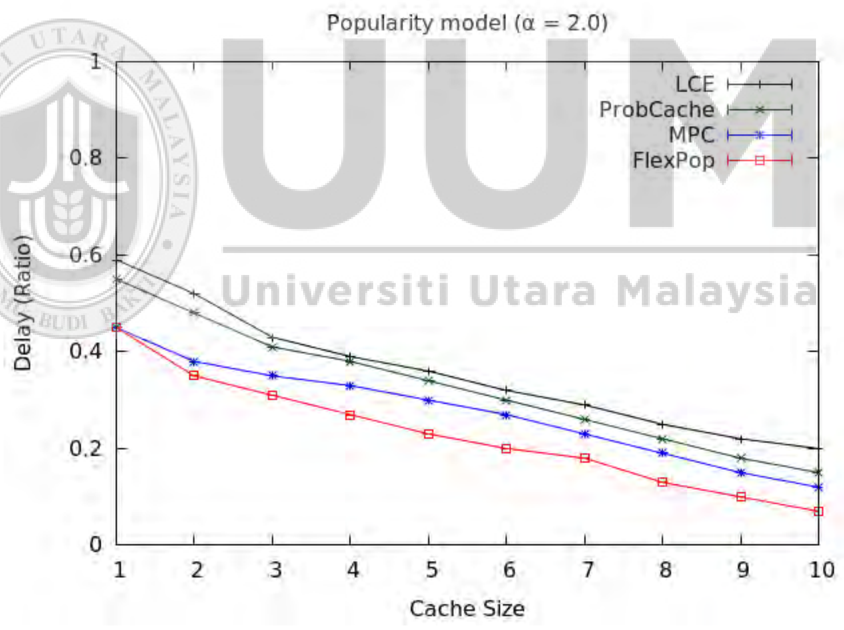


Figure 5.13. Content Retrieval Delay with $\alpha = 2.0$

12%, and FlexPop = 7%, respectively. The numerical result is presented in Table 5.4.

Hence, it guarantees that with high popularity scenarios (and even with low cache sizes), the content retrieval delay can be significantly minimized.

Table 5.4
Content Delay Comparison

Popularity model α				**Cache Size**		**Content Retrieval Delay**			
0.7	**1.0**	**1.5**	**2.0**	**100**	**1,000**	**LCE**	**ProbCache**	**MPC**	**FlexPop**
$\surd$	-	-	-	$\surd$	-	0.97	0.97	0.97	0.97
$\surd$	-	-	-	-	$\surd$	0.95	0.94	0.94	0.94
-	$\surd$	-	-	$\surd$	-	0.97	0.96	0.96	0.95
-	$\surd$	-	-	-	$\surd$	0.48	0.47	0.42	0.31
-	-	$\surd$	-	$\surd$	-	0.72	0.72	0.68	0.58
-	-	$\surd$	-	-	$\surd$	0.30	0.24	0.18	0.10
-	-	-	$\surd$	$\surd$	-	0.59	0.55	0.45	0.45
-	-	-	$\surd$	-	$\surd$	0.20	0.15	0.12	0.07

5.2.4 Memory Utilization

Memory utilization is also considered an important metric for analyzing the performance of ICN in terms of content retrieval delay. That is, if the memory is full and the new request arrives for a content but the content is not cached locally, then the request is forwarded to the server and therefore takes more time in accessing. The percentage of total memory utilization (TU_m) can be expressed by the following formula [219]:

$$TU_m = \frac{U_m}{T_m} * 100 \tag{5.6}$$

where U_m is the utilized memory by cached contents and T_m is the total available cache size.

In case the total available cache size is 10GB and the utilized memory is 3GB, put it in equation 5.6, we get

$$TU_m = \frac{3}{10} * 100 = 30\%. \tag{5.7}$$

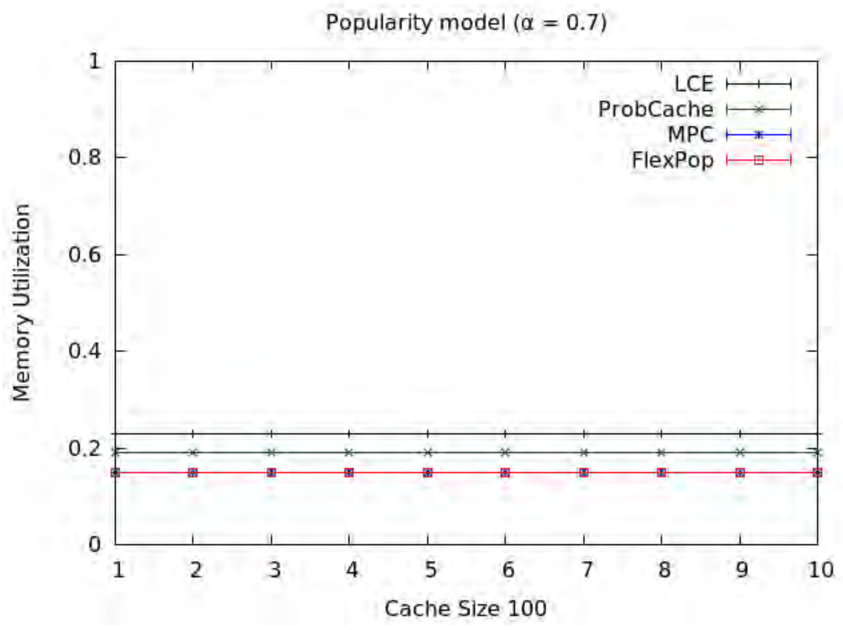


Figure 5.14. Utilized Memory with Cache Size 100 and $\alpha = 0.7$

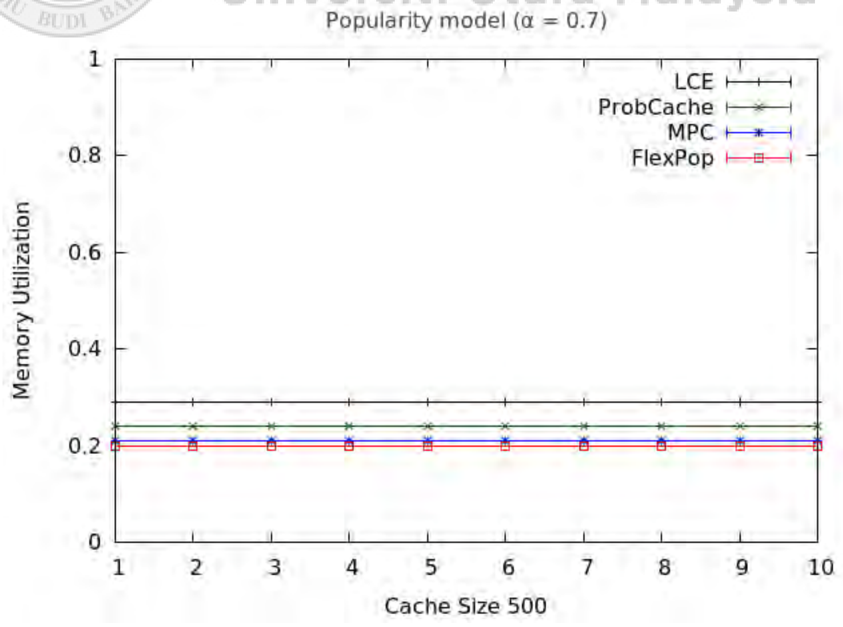


Figure 5.15. Utilized Memory with Cache Size 500 and $\alpha = 0.7$

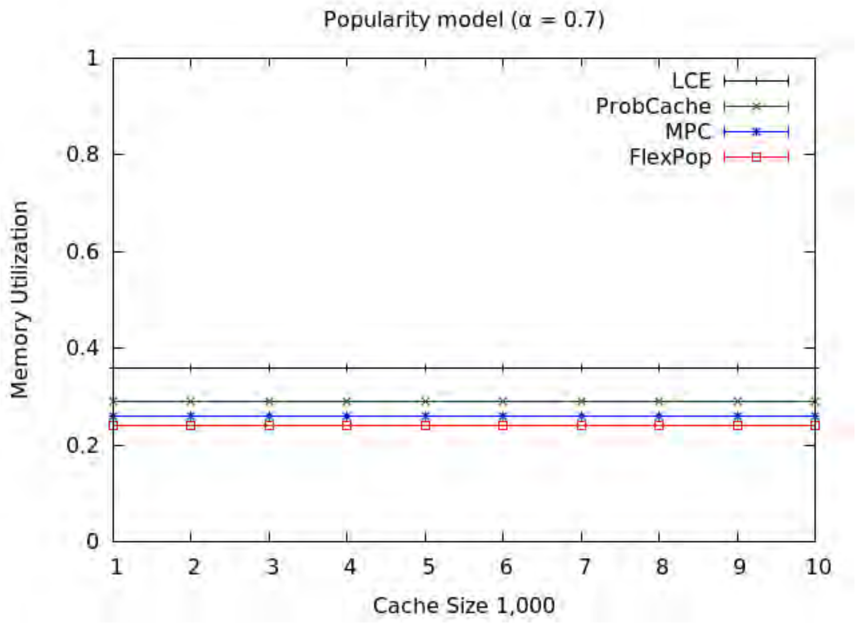


Figure 5.16. Utilized Memory with Cache Size 1,000 and $\alpha = 0.7$

Here we compare the performance of our proposed strategy with that of LCE, ProbCache, and MPC. In all figures, i.e., Figures 5.14 to 5.19, the x-axis shows the number of simulation runs and the y-axis shows the percentage of utilized memory. Figure 5.14 shows the average percentage usage of memory for all network nodes. The scenarios have been tested in OPNET Modeler 18.0 using Abilene topology with Facebook [165, 166, 167] against cache size 100, 500, and 1,000 elements, respectively. In the evaluated scenarios, the α parameter of the Zipf probability distribution is 0.7 and 2.0. In low popularity scenario ($\alpha = 0.7$) the percentage of utilized memory for LCE and ProbCache was recorded 23% and 19% for cache size 100, respectively; while there was a slight improvement in MPC and FlexPop, which was almost the same, i.e., 15% (see Figure 5.14). As the cache size was increased from 100 to 500 and 1,000 elements, LCE utilized 29% and 36% memory and ProbCache 24% and 29% memory with cache size 500 and 1,000, respectively. However, FlexPop got almost the same result as MPC, i.e., for cache size 500, MPC utilized 21% and FlexPop 20% memory, and for cache size 1,000, MPC used 26% while it was recorded 24% for FlexPop, as shown in Figures 5.15 and 5.16, respectively. Furthermore, with the increase of α pa-

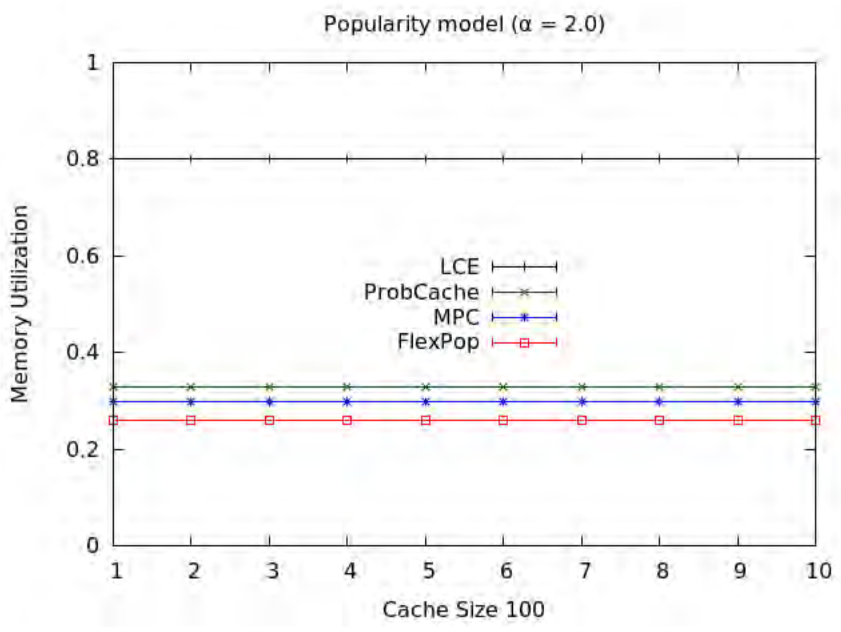


Figure 5.17. Utilized Memory with Cache Size 100 and $\alpha = 2.0$

rameter from 0.7 to 2.0, FlexPop outperformed all strategies in all cache sizes. Refer to Figure 5.17, with cache size 100, the utilization of memory was recorded as follows: LCE = 80%, ProbCache = 33%, MPC = 30%, and FlexPop = 26%. With cache size 500, LCE used 95% memory, ProbCache utilized 47%, MPC was recorded with 42%, and FlexPop was improved with 37%, as shown in Figure 5.18. With cache size 1,000, as demonstrated in Figure 5.19, the usage of memory was recorded as follows: LCE = 99%, ProbCache = 74%, MPC = 68%, and FlexPop = 48%.

Thus, it showed that with high popularity scenarios and higher cache sizes, the memory utilization can be noticeably minimized using FlexPop and therefore content retrieval delay can be significantly reduced. Table 5.5 shows the detail result of memory utilization in all four strategies.

5.2.5 Stretch

Stretch is also one of the most used metrics for calculating *Hop decrement* [41, 164, 220, 174, 122, 163, 58], represented by equation 5.8, and is referred to the ratio of

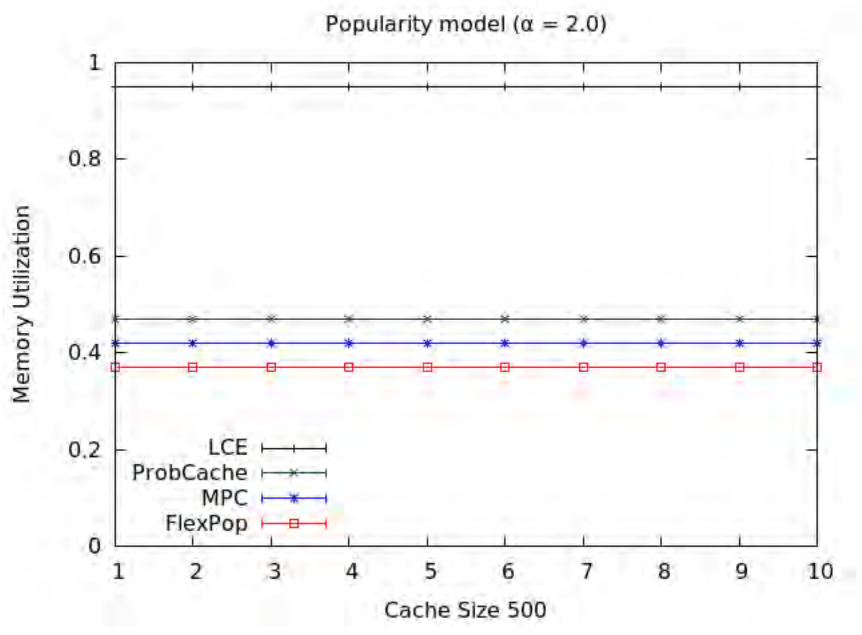


Figure 5.18. Utilized Memory with Cache Size 500 and $\alpha = 2.0$

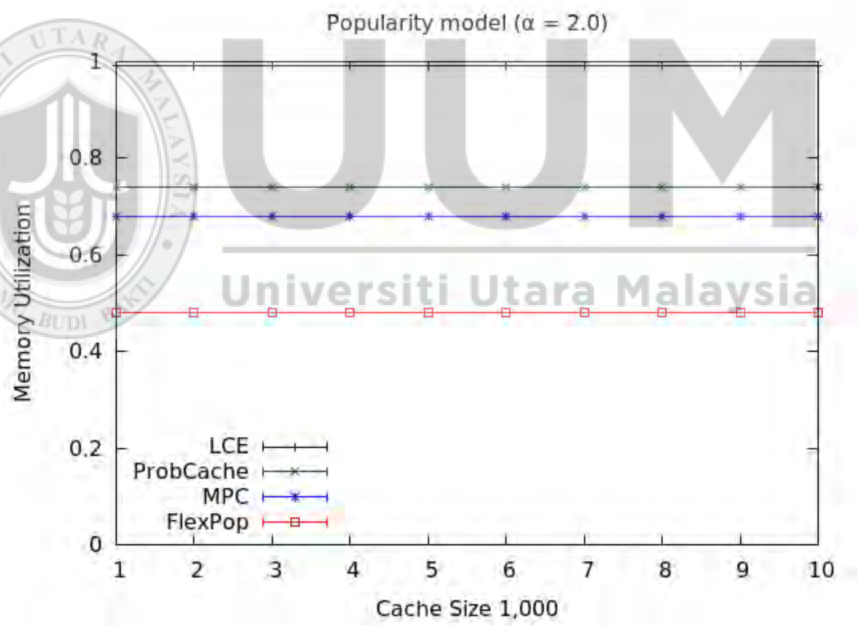


Figure 5.19. Utilized Memory with Cache Size 1,000 and $\alpha = 2.0$

complete path traversed from client to server by *content request.* Whereas *Hop decrement*, represented by equation 5.9, is the percentage of *content request hops* and the *path hops* between client and server. *Content delay* is also affected by *Hop decrement,* i.e., the lower the *Hop decrement* the higher the *content delay*, and vise versa.

Table 5.5
Memory Utilization Comparison

Popularity model α		Cache Size			Memory Utilization			
0.7	**2.0**	**100**	**500**	**1,000**	**LCE**	**ProbCache**	**MPC**	**FlexPop**
$\surd$	-	$\surd$	-	-	23	19	15	15
$\surd$	-	-	$\surd$	-	29	24	21	20
$\surd$	-	-	-	$\surd$	36	29	26	24
-	$\surd$	$\surd$	-	-	80	33	30	26
-	$\surd$	-	$\surd$	-	95	47	42	37
-	$\surd$	-	-	$\surd$	99	74	68	48

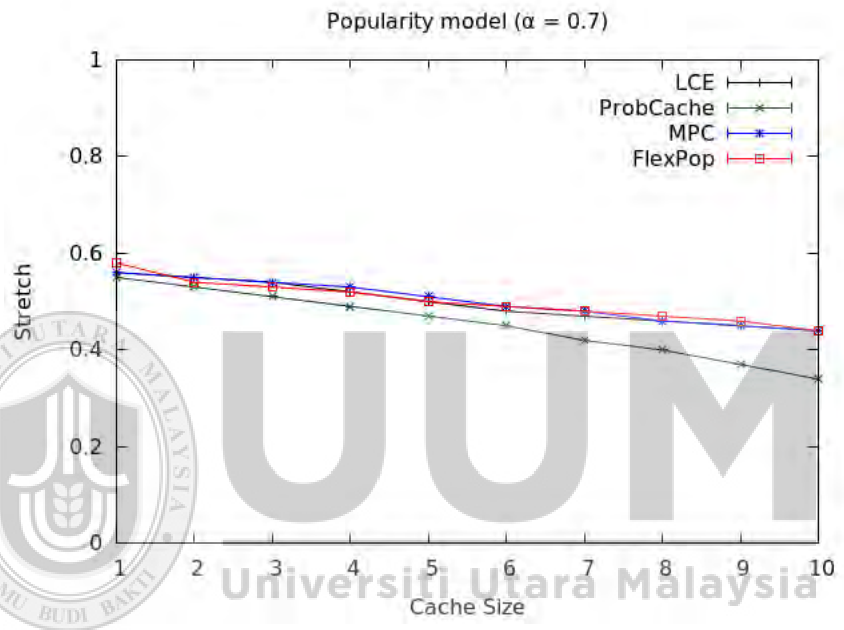


Figure 5.20. Stretch with $\alpha = 0.7$

$$S = \frac{\sum_{i=1}^{n} \mathscr{H}_c}{\mathscr{H}_t} \tag{5.8}$$

$$\mathscr{H}_d = 1 - S \tag{5.9}$$

where $\mathscr{H}$ is the number of hops traversed by content c and t is the number of total hops. S represents the Stretch and $\mathscr{H}_d$ is the hop decrement.

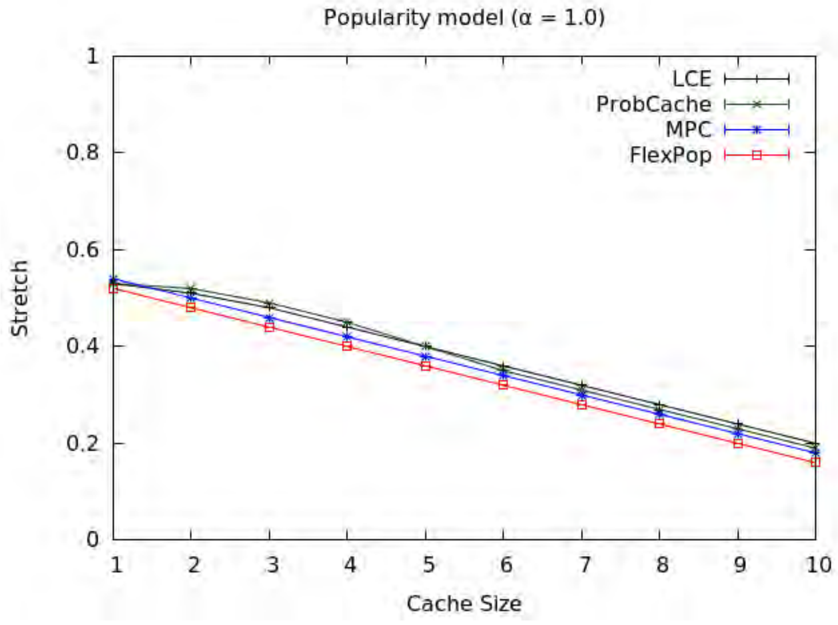


Figure 5.21. Stretch with $\alpha = 1.0$

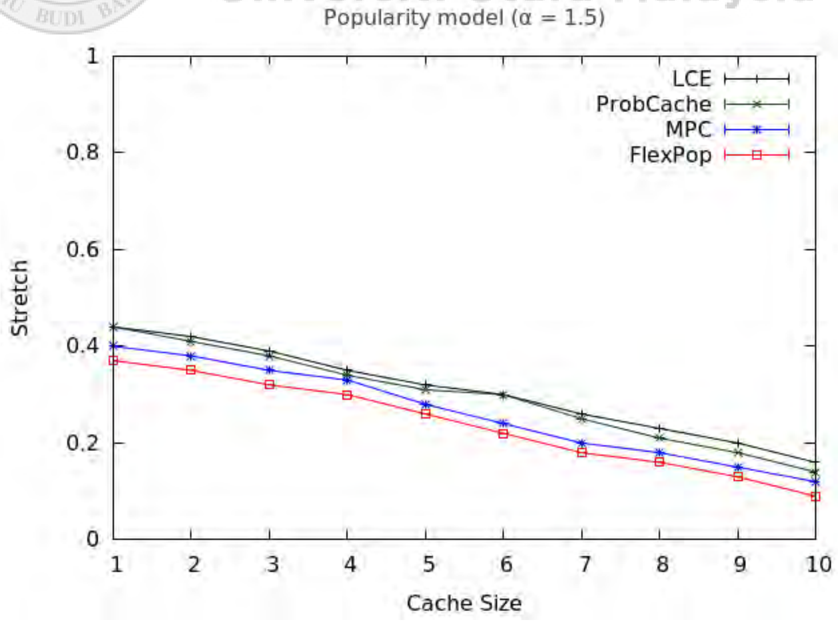


Figure 5.22. Stretch with $\alpha = 1.5$

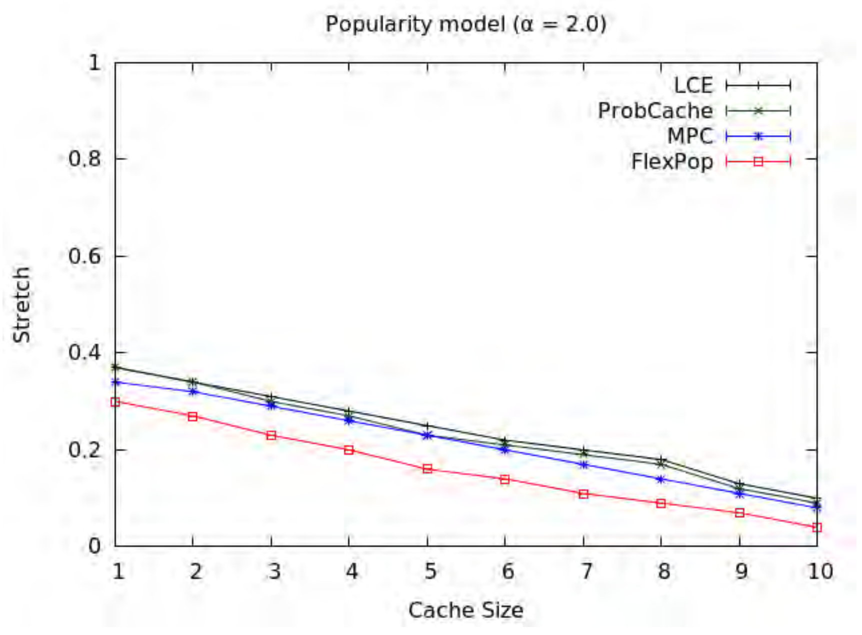


Figure 5.23. Stretch with $\alpha = 2.0$

Figures 5.20 to 5.23 show the percentage of stretch exhibited by LCE, ProbCache, MPC, and FlexPop on Abilene topology. As ProbCache tries to cache contents near to the subscriber, the stretch is somehow lower than LCE, MPC, and FlexPop with low popularity model and lower cache size (see Figure 5.20). However, when the cache size is increased from 100 to 1,000 elements, all strategies achieved almost the same amount of stretch.

Similarly, when the popularity model α was increased from 0.7 to 1.0, the stretch ratio was somehow the same with cache size 100 in all strategies, but with the increase of cache size, MPC, ProbCache, and FlexPop achieved equivalent result, a little lower than LCE, as shown in Figure 5.21. Moreover, with $\alpha = 2.0$, presented in Figure 5.23, FlexPop achieved the best result in all cache sizes, which was 37%-10% in LCE, 37%-9% in ProbCache, 34%-8% in MPC, and 30%-4% in FlexPop with cache size 100-1,000 elements, respectively. The complete result of stretch ratio is given in Table 5.6.

Table 5.6
Stretch Comparison

Popularity model α				Cache Size		Stretch			
0.7	**1.0**	**1.5**	**2.0**	**100**	**1,000**	**LCE**	**ProbCache**	**MPC**	**FlexPop**
$\surd$	-	-	-	$\surd$	-	0.56	0.55	0.56	0.58
$\surd$	-	-	-	-	$\surd$	0.44	0.34	0.44	0.44
-	$\surd$	-	-	$\surd$	-	0.56	0.53	0.54	0.54
-	$\surd$	-	-	-	$\surd$	0.22	0.19	0.21	0.20
-	-	$\surd$	-	$\surd$	-	0.44	0.44	0.40	0.37
-	-	$\surd$	-	-	$\surd$	0.16	0.14	0.12	0.09
-	-	-	$\surd$	$\surd$	-	0.37	0.37	0.34	0.30
-	-	-	$\surd$	-	$\surd$	0.10	0.09	0.08	0.04

5.3 Topological Effect on Simulation Results

As stated earlier [174] that topology has direct impact on the simulation results and according to the ICN baseline scenarios [16, 15], there is no general agreement on the topology selection, for the fair evaluation we have also simulated all strategies on GEANT and DTelekom topologies. To evaluate the performance of ICN caching, we believe that out of our selected five metrics, cache hit and stretch ratio are considered the most prominent because cache hit combines the properties of other metrics, such as content eviction rate, cached element rate, and cached miss rate, while stretch ratio also affects the hop decrement as well as content retrieval delay. Therefore, we present simulation results for the cache hit rate and stretch ratio on the mentioned two topologies in the following subsections.

5.3.1 Cache Hit on GEANT and DTelekom

The ratio of cache hit on GEANT and DTelekom topologies is shown in Figure 5.24 and 5.25, respectively. In low popularity scenario, i.e., when α is 0.7, the ratio of cache hit was low on both GEANT and DTelekom topologies with cache sizes 100 to 1,000 elements (1GB to 10GB), but with the increase of cache size as well as the increase of popularity value, especially with $\alpha = 2.0$, which represents multimedia applications, the higher hit ratio was achieved. On the GEANT topology, the recorded

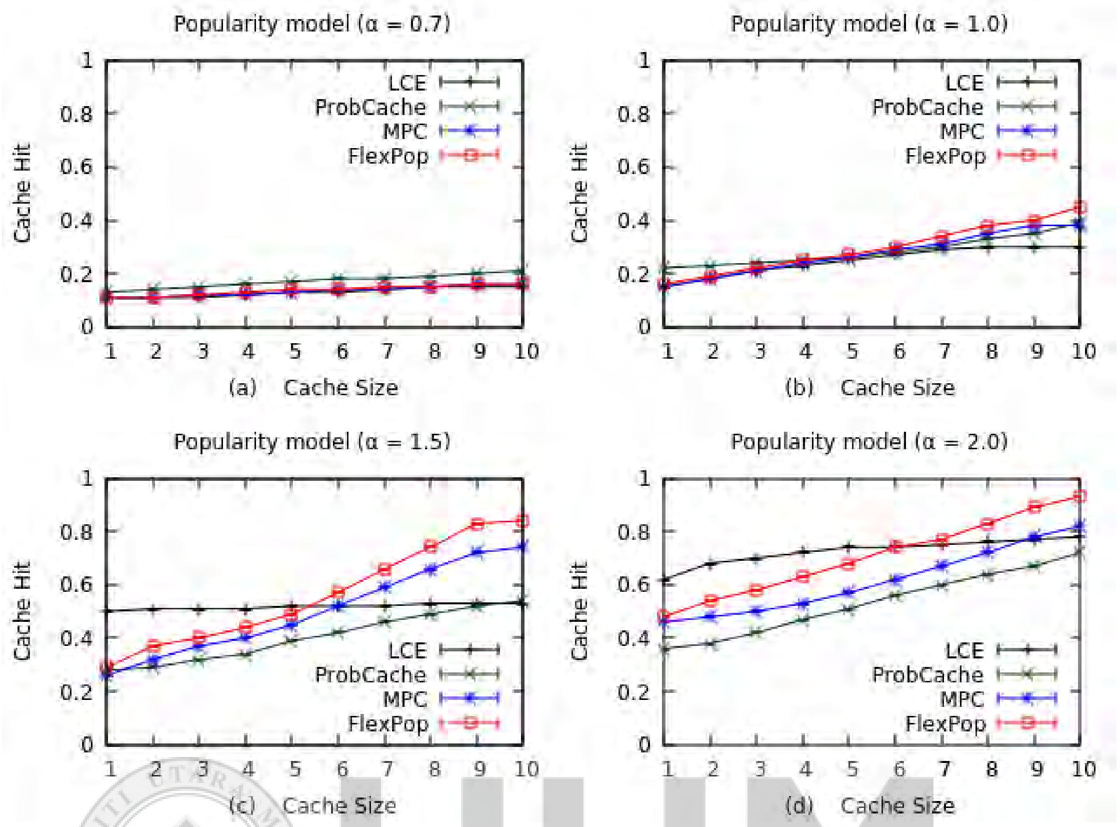


Figure 5.24. Cache Hit Rate on GEANT Topology

hit ratio was 15%, 16%, 16%, and 21% for LCE, MPC, FlexPop, and ProbCache, respectively, with popularity value 0.7. However, with the increase of popularity model the hit ratio reached 45% with $\alpha = 1.0$ in FlexPop as compared to 38% of MPC, 39% of ProbCache, and 30% of LCE; 84% with $\alpha = 1.5$ compared to 74% of MPC, 54% of ProbCache, and 53% of LCE; and 93% with $\alpha = 2.0$ in comparison with 82% of MPC, 72% of ProbCache, and 78% of LCE, respectively. Similarly, on the DTelekom topology, the achieved simulated results for $\alpha = 2.0$ (multimedia applications) were as follows: LCE =78%, which is the same as on GEANT topology, ProbCache = 77%, MPC = 91%, and FlexPop = 99%. The overall result for the remaining parameters is presented on Tables 5.7 and 5.8 for the GEANT and DTelekom topologies, respectively.

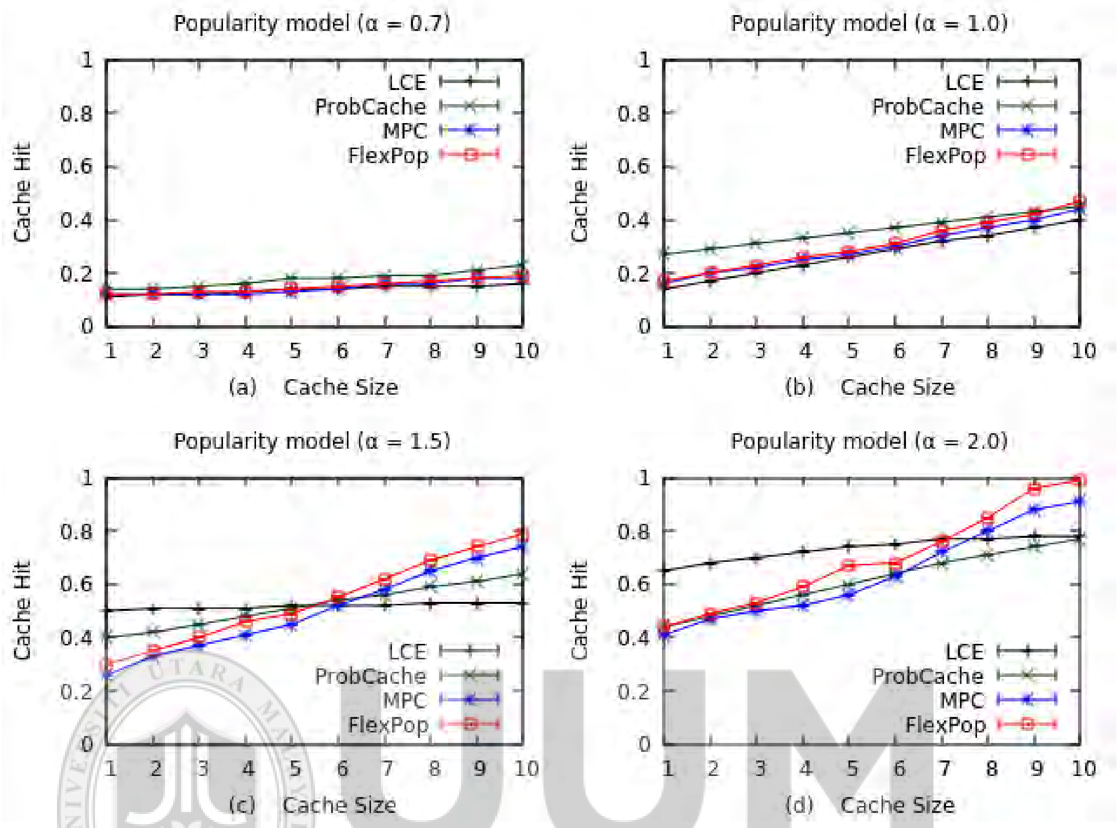


Figure 5.25. Cache Hit Rate on DTelekom Topology

Table 5.7
Cache Hit Comparison on GEANT Topology

Popularity model α				**Cache Size**		**Cache Hit**			
0.7	**1.0**	**1.5**	**2.0**	**100**	**1,000**	**LCE**	**ProbCache**	**MPC**	**FlexPop**
√	-	-	-	√	-	0.11	0.13	0.11	0.11
√	-	-	-	-	√	0.15	0.21	0.16	0.16
-	√	-	-	√	-	0.15	0.22	0.15	0.16
-	√	-	-	-	√	0.30	0.39	0.38	0.45
-	-	√	-	√	-	0.50	0.28	0.26	0.29
-	-	√	-	-	√	0.53	0.54	0.74	0.84
-	-	-	√	√	-	0.62	0.36	0.46	0.48
-	-	-	√	-	√	0.78	0.72	0.82	0.93

Table 5.8
Cache Hit Comparison on DTelekom Topology

Popularity model α				**Cache Size**		**Cache Hit**			
0.7	**1.0**	**1.5**	**2.0**	**100**	**1,000**	**LCE**	**ProbCache**	**MPC**	**FlexPop**
$\surd$	-	-	-	$\surd$	-	0.11	0.14	0.12	0.12
$\surd$	-	-	-	-	$\surd$	0.16	0.23	0.18	0.19
-	$\surd$	-	-	$\surd$	-	0.14	0.27	0.16	0.17
-	$\surd$	-	-	-	$\surd$	0.40	0.45	0.44	0.47
-	-	$\surd$	-	$\surd$	-	0.50	0.40	0.26	0.30
-	-	$\surd$	-	-	$\surd$	0.53	0.64	0.74	0.79
-	-	-	$\surd$	$\surd$	-	0.65	0.44	0.41	0.44
-	-	-	$\surd$	-	$\surd$	0.78	0.77	0.91	0.99

5.3.2 Stretch on GEANT and DTelekom

Figures 5.26 and 5.27 show the stretch ratio exhibited by LCE, ProbCache, MPC, and FlexPop on the GEANT and DTelekom topology, respectively. The stretch was almost the same for all strategies with low popularity model and lower cache size on both topologies. When the popularity model α was increased from 0.7 to 1.0, the stretch ratio was somehow the same in all strategies with all cache sizes. Moreover, with $\alpha = 2.0$, the recorded stretch ratio was different on both topologies, i.e., on the GEANT topology it ws 45%-18% in LCE with cache size 100-1,000 elements, 41%-16% in ProbCache, 40%-14% in MPC, and 34%-10% in FlexPop. While on the DTelekom topology the recorded stretch with the same parameters was as follows: LCE =37%-10%, ProbCache =34%-7%, MPC =34%-08%, and FlexPop =30%-4%, respectively. The result for the remaining parameters on both GEANT and DTelekom topologies is presented on Tables 5.9 and 5.10, respectively.

5.4 Discussion

Throughout this chapter the performance of FlexPop is assessed with respect to different parameters, such as popularity model variation, cache size, metrics measurement, and topological impact, i.e., cache deployment. As discussed earlier that FlexPop is designed for measuring the performance of multimedia applications, it does not per-

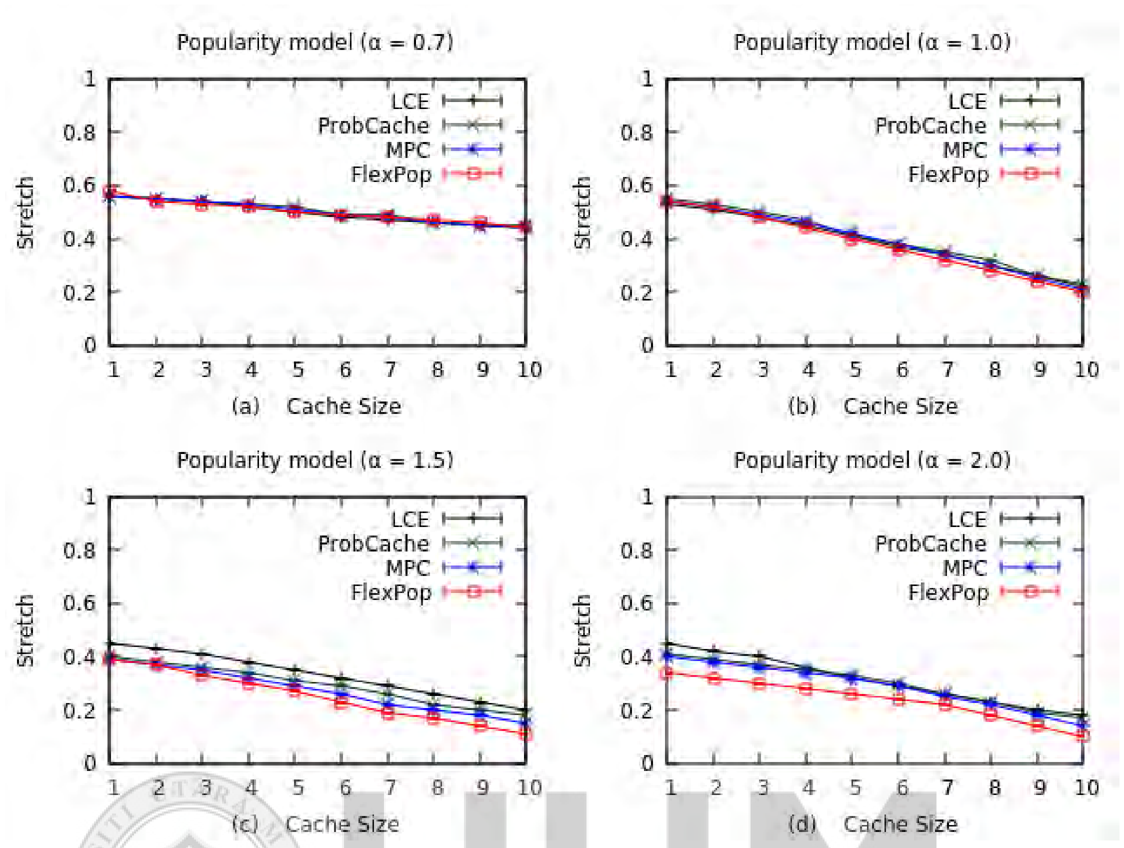


Figure 5.26. Stretch Ratio on GEANT Topology

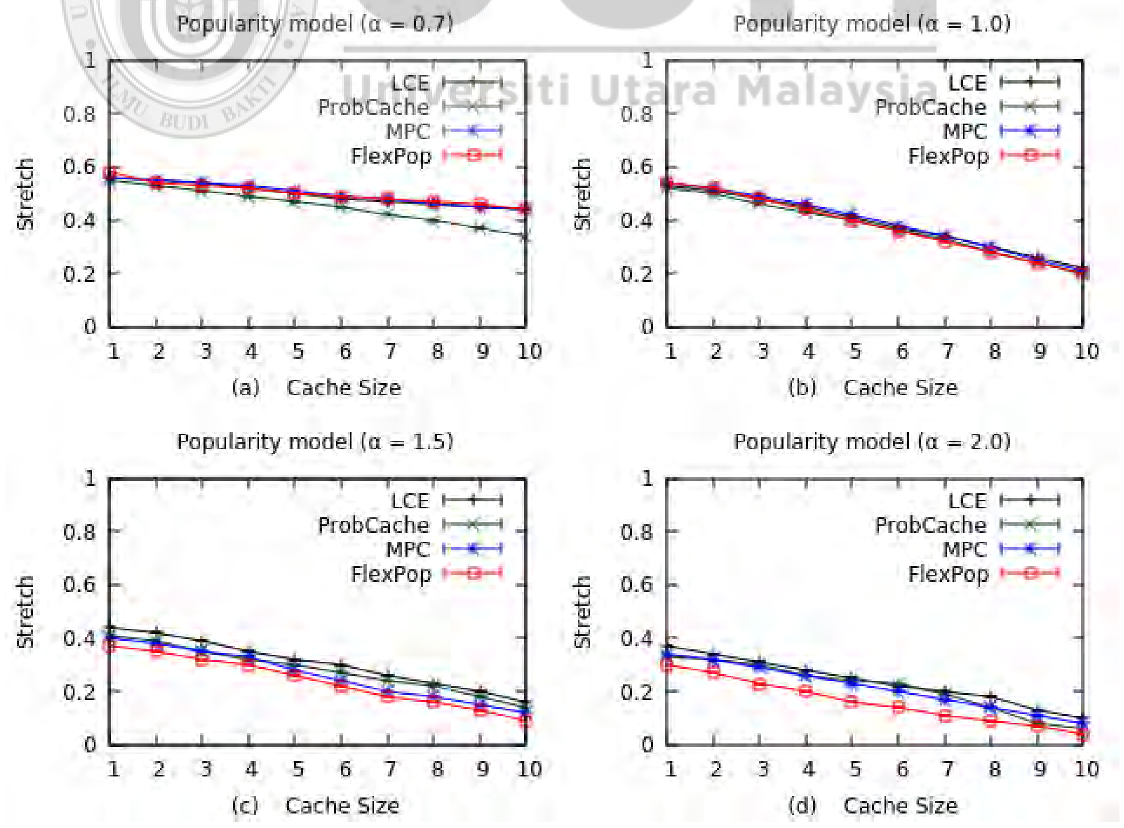


Figure 5.27. Stretch Ratio on DTelekom Topology

Table 5.9
Stretch Comparison on GEANT Topology

Popularity model α				**Cache Size**		**Stretch Ratio**			
0.7	**1.0**	**1.5**	**2.0**	**100**	**1,000**	**LCE**	**ProbCache**	**MPC**	**FlexPop**
√	-	-	-	√	-	0.56	0.56	0.56	0.58
√	-	-	-	-	√	0.44	0.44	0.44	0.44
-	√	-	-	√	-	0.54	0.54	0.54	0.54
-	√	-	-	-	√	0.22	0.21	0.21	0.20
-	-	√	-	√	-	0.45	0.42	0.42	0.39
-	-	√	-	-	√	0.20	0.18	0.15	0.11
-	-	-	√	√	-	0.45	0.41	0.40	0.34
-	-	-	√	-	√	0.18	0.16	0.14	0.10

Table 5.10
Stretch Comparison on DTelekom Topology

Popularity model α				**Cache Size**		**Stretch Ratio**			
0.7	**1.0**	**1.5**	**2.0**	**100**	**1,000**	**LCE**	**ProbCache**	**MPC**	**FlexPop**
√	-	-	-	√	-	0.56	0.55	0.56	0.58
√	-	-	-	-	√	0.44	0.34	0.44	0.44
-	√	-	-	√	-	0.56	0.49	0.54	0.54
-	√	-	-	-	√	0.22	0.21	0.21	0.20
-	-	√	-	√	-	0.46	0.42	0.42	0.40
-	-	√	-	-	√	0.16	0.14	0.12	0.09
-	-	-	√	√	-	0.37	0.34	0.34	0.30
-	-	-	√	-	√	0.10	0.07	0.08	0.04

form well with low popularity model. The main reason for this is as the skewness of the distribution (which is measured through α) increases, popular contents attain maximum portion of the Internet traffic. And it is also accepted that multimedia applications (i.e., VoD) only from YouTube and Netflex attract more than half of the overall traffic [13]. Thus, with high popularity model, the FlexPop achieves better results. The technical factor for the LCE path redundancy is such that LCE caches every incoming content at every node on the publisher-subscriber path, thus path redundancy is 100%.

In addition, the current available DRAM is 10GB (maximum), therefore, after some times the memory becomes full and content eviction starts for the accommodation of new arrived contents. Consequently, the cache hit rate is reduced and subsequent requests for the evicted contents are forwarded to the server, which increases content retrieval delay. To add more, with the eviction of contents, hop decrement is affected and thus stretch ratio increases. Similarity, MPC and ProbCache also cache diverse kinds of contents (as they are not particularly designed for multimedia applications) and the popular ones (i.e., VoD) will not have enough space to be cached. Moreover, in case of memory overflow, there is no eviction policy in ProbCahe and MPC but they simply delete the contents which affect the mentioned metrics.

Cache size is another factor of the FlexPop result supremacy, i.e., when the network becomes stable then LCE, ProbCache, and MPC cache sizes overflow and thus eviction operation starts. On the other hand, unlike LCE, FlexPop does not cache every content and hence still have space for accommodating new contents rather than deleting the cached ones. Besides, MPC and ProbCache store lesser contents than LCE but because of having no eviction policy for MPC and ProbCache, the popular contents are evicted and subsequent requests for those contents are forwarded to the original server. This leads to reduce the performance of MPC and ProbCache. While in FlexPop the contents are not deleted but cached at the other router having second number of highest outgoing interfaces.

Another factor of the FlexPop dominance is the cache deployment mechanism. That is, in FlexPop the contents are cached at the router which has the highest number of outgoing interfaces, therefore, large portion of the traffic passes through that router. This makes FlexPop different from LCE, ProbCache, and MPC in redundant content caching, which in turn increases cache hit rate and reduces extra retrieval delay. Moreover, when the contents are evicted in the FlexPop, they are cached at another location which is also nearer to the user(s). In other words, this mechanism increases the hop decrement which consequently reduces the stretch ratio.

5.5 Summary

This chapter discussed the performance evaluation of FlexPop in comparison with LCE, ProbCache, and MPC caching strategies designed for ICN. The proposed strategy was implemented in two simulators, i.e., for the evaluation of cache hit rate, redundancy, and content retrieval delay in SocialCCNSim, and in OPNET Modeler 18.0 for measuring the utilization of memory. Finally, the obtained results showed that the FlexPop achieved better performance than LCE, ProbCache, and MPC in cache hit, redundancy elimination, content retrieval delay, memory utilization, and stretch. Consequently, the FlexPop proved that it can be the accurate choice for considering caching strategy in the future Internet.

CHAPTER SIX
CONCLUSION AND FUTURE WORK

The main goal of this thesis is to design a Flexible Popularity-based Caching (FlexPop) strategy for Information-Centric Networking (ICN). During this course of research, we discovered challenges for cross-comparison of the caching contributions for ICN. In the literature, we studied a handful number of simulation environments and parameters ranging from popularity models to topologies, and then based on the draft of Internet Engineering Task Force (IETF) [15], simulation parameters were carefully selected. The performance of FlexPop strategy was extensively evaluated through simulation in two network simulators, i.e., SocialCCNSim and OPNET Modeler 18.0. As discussed in Chapter Four, for the validation of FlexPop, we tested our work on four ISP-level topologies, i.e., Abilene, Tiger, GEANT, and DTelekom, however, for the remaining parameters the simulations were run only on Abilene topology. The chosen catalog size was 10^6 elements and the cache size ranging form 100 to 1,000 elements. For popularity, we chose four values of α for the Zipf model, i.e., 0.7, 1.0, 1.5, and 2.0. The remaining chapter is organized as follows:

The importance of FlexPop strategy for the future Internet with its possible implementation is discussed in Section 6.1. The research contributions are presented in Section 6.2, while the limitations are highlighted in Section 6.3. Finally, some suggestions are given in Section 6.4 for future studies.

6.1 Research Summary

The existing Internet architecture follows source-driven approach, where source needs to establish a connection with receiver before sending any data. This approach reduces the ability of fully resource utilization that are available all the way from the publisher (receiver) to the subscriber (user). To overcome this problem, Van Jacobson [30] proposed an idea in which the customer requests content without knowing the providing host and the path is established by the publisher to the subscriber. In other words, the approach follows receiver-driven approach and the data follows the reverse path. The system then takes the responsibility of mapping the requested data and its location. This approach is called Information-Centric Networking (ICN).

As presented in Chapter One, the ICN network is classified into a multi-stage caching system, where the requests are generated from geographically dispersed nodes and the caches are physically located across the network. Now, if the requested content is available somewhere cached by a network node, the request is satisfied locally. This leads to the problem of effective caching strategies for content placement and replication. The main goal of this thesis was to introduce a new caching strategy for content placement and eviction to deal with the mentioned problems.

Even though, many strategies have been designed for this reason, as discussed in Chapter Two, however, it was observed and concluded that the existing strategies cannot handle the challenges caused by content placement and eviction. For example, LCE, which is the default-ICN strategy, increases content redundancy, while ProbCache and MPC have no eviction policy to accommodate the new arrived contents and thus the popular contents are replaced with the new arrived ones, which reduce the cache hit rate and in turn increase content retrieval delay. It motivated and prompted this research work to design a new caching strategy for ICN called Flexible Popularity-based Caching (FlexPop).

To achieve the objectives of this research, DRM was followed as a guideline and framework, and network modeling and simulation process was adopted as proposed in

[[138, 144]] and discussed in Chapter Three. Moreover, two mechanisms of FlexPop strategy, i.e., Content Placement Mechanism (CPM), which is used for popular content placement, and Content Eviction Mechanism (CEM) designed for content eviction in a flexible manner, were introduced in Chapter Four.

Simulation results showed that FlexPop strategy can increase cache hit rate, utilize memory efficiently, and reduce redundancy, stretch, and content retrieval delay, which were presented intensively in Chapter Five.

In achieving the performance evaluation of FlexPop strategy, the simulation results confidently emphasize that the main objectives of this study have been fully and successfully accomplished.

6.2 Research Contributions

The vital contribution of this research was to design a FlexPop strategy to improve performance of the future Internet. Moreover, two mechanism, i.e., CPM and CEM were proposed for achieving better results of FlexPop strategy.

The specific contributions consist of the following points:

1. The development of FlexPop strategy for the future Internet that can serve as a benchmark for any intended improvement of ICN caching.
2. The design of a new ICN caching strategy for multimedia applications that leads to the following applicable contributions:
    a) Efficiently increase cache hit rate of the requested content items, which is more effective for data requests of frequently accessed objects.

   b) Reduce path redundancy during content placement, which results in a better memory management.
   c) Minimize searching overhead during content eviction and increase hop decrement that lead to reduced latency and stretch in content delivery.

3. The adaptation of caching was made by developing two new mechanisms for content placement and eviction to improve caching gain and simplify content access.
4. *Simple Additive Weighting* (SAW), which is one of the *Multiple Attribute Decision Making* (MADM) approaches and based on *Fuzzy Set Theory*, was adapted for content placement to obtain acceptable results in terms of resource utilization.
5. Content eviction is based on another MADM approach called *Technique for Order Preference by Similarity to Ideal Solution* (TOPSIS), where the evicted content is cached at a router that has a shorter distance from the subscriber.
6. The verification and validation of the proposed strategy was achieved by implementing it in SocialCCNSim and OPNET Modeler 18.0 simulators and comparing the obtained results with other caching strategies, i.e., LCE, ProbCache, and MPC.

A summary of the research contributions is demonstrated in Figure 6.1. In the first phase, the CPM mechanism was modeled following the fuzzy SAW approach for content placement. In the next phase, using fuzzy TOPSIS method the CEM mechanism was modeled to flexibly evict the contents and cache them at other location that would be nearer to the user(s). Finally, the FlexPop strategy was simulated and the results were compared with those of LCE, ProbCache, and MPC in the SocialCCNSim simulator for cache hit rate, redundancy, content retrieval delay and stretch, while for memory utilization the simulations were conducted using OPNET Modeler 18.0.

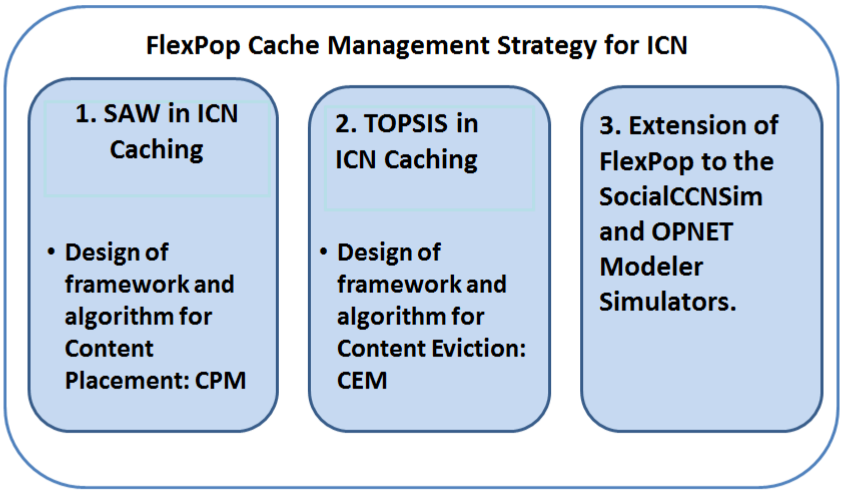

FlexPop Cache Management Strategy for ICN
1. SAW in ICN Caching
• Design of framework and algorithm for Content Placement: CPM
2. TOPSIS in ICN Caching
• Design of framework and algorithm for Content Eviction: CEM
3. Extension of FlexPop to the SocialCCNSim and OPNET Modeler Simulators.

Figure 6.1. Research Contributions

6.3 Research Limitations

Despite the fact that this research was conducted under careful determination and use of an arrangement of supporting schemes and guidelines, it is constrained to some particular applications. Most importantly, in the simulations conducted in OPNET Modeler 18.0, the processors used for packet sending and receiving are IP processors, as there is no specific processor designed for ICN so far, there might be a slight difference in the simulation results if it is run on the real ICN processors.

Furthermore, the proposed strategy performs well in the infrastructure-based (fixed) wireless networks, however, it will be difficult to specify the router with maximum outgoing interfaces if it is applied on every ad hoc networks, as in [106]. In addition, if two cached contents, for example *A* and *B*, have the same popularity value and access time, then it is very tough to decide which content should be evicted to accommodate new incoming content. For this process, the decision is left on LRU, i.e., either to evict *A* or *B*, however, due to the same popularity value and access time, this eviction process will not affect the overall result.

6.4 Future Directions

Caching and replication mechanisms have been extensively thought-out at the application level, mostly in the context of web applications. It has been recently studied [221] that the advantages from the widespread utilization of caching in ICN will not be astonishing. Although they raise significant concerns about the operation of the envisioned caching procedures, these analyses are generally based on research performed more than a decade ago [222]. Further research on existing Internet architecture could shed additional light on the importance of information today and thus to the achievable advantages from widespread caching. For example, a recent study [223] has shown that during the past few years application level caching performance has been effectively changed by web information popularity. Therefore, understanding the implications of cache space may help in improving customer experience with regard to bandwidth utilization.

Another issue is that when content items are cached inside the network, various traffic types will compete for the same caching space, therefore, cache management becomes crucial for the network. Recent studies on cache management have shown that intelligent strategies can significantly improve the performance of ICN [41, 106, 118].

Moreover, on-path caching is a straightforward approach to content placement as they move from source to destination. The communication and computational complexity can be reduced using on-path caching but, conversely, the chances of hitting cached content items might also be reduced. In addition, mechanisms for storing content items inside the network open up the possibility of in-network cache management, routing and forwarding. For example, cache locations, indications of cache contention and cache-ability of information can affect routing decisions.

Staleness detection of cached NDOs can be another challenging task in ICN caching [224, 66]. As copies of NDOs are largely distributed in in-network caches, ICN must have the capability of providing staleness verification procedure for synchronization of NDOs placed at their publishers and in-network caching nodes. To handle this issue, two types of approaches, namely direct and indirect approaches, can be considered. Direct approach is suitable for some NDOs because in this approach each cache searches specific information in the name of NDOs, for example, time stamp which directly specifies its staleness. In the indirect approach, the provider of the cached NDO is consulted about its staleness before providing it. This approach is well applicable to those NDOs for which it is difficult to set their expiring time in advance, for example, a web page that includes the main text (which remains the same) and the interactive portion (which does not need regular revision) such as ads and comments.

The main work presented in this thesis can be extended in several ways, for example, the FlexPop strategy is tested only for multimedia applications, however, it can be modified and used for real time applications. Moreover, it would also be interesting to improve the proposed CPM mechanism if the caches of routers (after sharing comparison table) become full. Similarly, the CPM and CEM mechanisms can be enhanced in such a way to take mobility into consideration if they are applied on every ad hoc networks, such as vehicular ad hoc networks (VANETs).

The performance of FlexPop was compared with LCE - the default ICN strategy, Prob-Cache - which produced better results than LCE [41], and MPC - which has outperformed most of the existing strategies [58]. However, it would likewise be worthy to examine its performance against other caching strategies in different network domains such as 5G network and cloud infrastructure. Furthermore, the overall simulation results are obtained on Abilene topology except for the cache hit rate, which has been examined on GEANT, Tiger, and DTelekom topologies. Nevertheless, it would also be useful to test the performance of FlexPop against other metrics on the remaining three topologies.

REFERENCES

[1] G. Xylomenos, C. N. Ververidis, V. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, G. C. Polyzos *et al.*, "A survey of Information Centric Networking research," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 2, pp. 1024–1049, 2014.

[2] A. Feldmann, "Internet clean-slate design: What and why?" *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 3, pp. 59–64, 2007.

[3] M. Handley, "Why the Internet only just works," *BT Technology Journal*, vol. 24, no. 3, pp. 119–129, 2006. [Online]. Available: http://www0.cs.ucl.ac.uk/staff/M.Handley/papers/only-just-works.pdf

[4] N. Fotiou, "Information-Centric Networking: Security requirements and solutions," Ph.D. dissertation, School of Information Sciences and Technology, Department of Computer Science, Ahens University of Economics and Business, 2014. [Online]. Available: http://phdtheses.ekt.gr/eadd/handle/10442/34673

[5] V. Jacobson, "A new way to look at networking," *Google Tech Talk*, vol. 30, 2006. [Online]. Available: https://github.com/dominictarr/cyphernet/issues/15

[6] D. Trossen, M. Sarela, and K. Sollins, "Arguments for an Information Centric Networking architecture," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 2, pp. 26–33, 2010.

[7] J. Rexford and C. Dovrolis, "Future Internet architecture: Clean-slate versus evolutionary research," *Communications of the ACM*, vol. 53, no. 9, pp. 36–40, 2010.

[8] Information-Centric Networking Research Group (ICNRG). [Online]. Available: http://trac.tools.ietf.org/group/irtf/trac/wiki/icnrg

[9] S. Carew, "Users complain, AT&T blames data tsunami," *Reuters Media File*, 2012. [Online]. Available: http://blogs.reuters.com/mediafile/2012/02/14/users-complain-att-blames-data-tsunami/

[10] V. Sourlas, "Replication Management and Cache Aware Routing in Information-Centric Networking," Ph.D. dissertation, Electrical & Computer Engineering, University of Thessaly, Greece, 2013.

[11] Cisco Visual Networking index: Forecast and methodology: 2013-2018, Tech. Rep., June, 2014. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.html

[12] Visual-Capitalist. What happens in an Internet minute in 2016. [Online]. Available: http://www.visualcapitalist.com/what-happens-internet-minute-2016/

[13] Intel. What happens in an Internet minute. [Online]. Available: http://www.intel.com/content/www/us/en/communications/internet-minute-infographic.html

[14] Internet Live Stats: Accessd: June 1, 2015, Tech. Rep. [Online]. Available: http://www.internetlivestats.com/

[15] K. Pentikousis, B. Ohlman, D. Corujo, G. Boggia, G. Tyson, E. Davies, A. Molinaro, and S. Eum, "Information-Centric Networking: Baseline scenarios," Tech. Rep., 2015. [Online]. Available: http://www.rfc-editor.org/info/rfc7476

[16] J. Seedorf, M. Arumaithurai, A. Tagami, K. Ramakrishnan, and N. Blefari-Melazzi, "Using ICN in disaster scenarios," Internet-Draft-work in progress 02, IETF, Tech. Rep., 2014.

[17] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of Information-Centric Networking," *Communications Magazine, IEEE*, vol. 50, no. 7, pp. 26–36, 2012.

[18] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, pp. 181–192, 2007.

[19] NSF Named Data Networking project. [Online]. Available: www.named-data.org/

[20] NSF Mobility First project. [Online]. Available: http://mobilityfirst.winlab.rutgers.edu/

[21] CCN Project. [Online]. Available: http://www.ccnx.org/

[22] P. Jokela, A. Zahemszky, C. Esteve Rothenberg, S. Arianfar, and P. Nikander, "LIPSIN: Line speed publish/subscribe inter-networking," in *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4. ACM, 2009, pp. 195–206.

[23] B. Ahlgren, M. D'Ambrosio, C. Dannewitz, A. Eriksson, J. Golic, B. Gronvall, D. Horne, A. Lindgren, O. Mammela, M. Marchisio *et al.*, "Second NetInf architecture description," 4ward eu fp7 project, deliverable d-6.2 v2. 0, apr. 2010, fp7-ict-2007-1-216041-4ward/d-6.2." [Online]. Available: http://www.4ward-project.eu/index.php?id=192

[24] B. Ahlgren, M. D'Ambrosio, M. Marchisio, I. Marsh, C. Dannewitz, B. Ohlman, K. Pentikousis, O. Strandberg, R. Rembarz, and V. Vercellone, "Design considerations for a network of information," in *Proceedings of the 2008 ACM CoNEXT Conference*. ACM, 2008, p. 66.

[25] FP7 Publish-Subscribe Internet Technology: PURSUIT project. [Online]. Available: http://www.fp7-pursuit.eu/PursuitWeb/

[26] FP7 CONVERGENCE project. [Online]. Available: http://www.ictconvergence.eu/

[27] FP7 Scalable and Adaptive Internet Solutions: SAIL project. [Online]. Available: http://www.sail-project.eu/

[28] FP7 COntent Mediator architecture for content-aware nETworks: COMET project. [Online]. Available: http://www.comet-project.org/

[29] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking (draft)," in *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum fÃ1/4r Informatik, 2011.

[30] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking Named Content," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*. ACM, 2009, pp. 1–12.

[31] C. Fang, F. R. Yu, T. Huang, J. Liu, and Y. Liu, "A survey of energy-efficient caching in Information-Centric Networking," *Communications Magazine, IEEE*, vol. 52, no. 11, pp. 122–129, 2014.

[32] N. Laoutaris, H. Che, and I. Stavrakakis, "The LCD interconnection of LRU caches and its analysis," *Performance Evaluation*, vol. 63, no. 7, pp. 609–634, 2006.

[33] C. Bernardini, T. Silverston, and O. Festor, "MPC: Popularity-based caching strategy for Content Centric Networks," in *Communications (ICC), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3619–3623.

[34] J. Li, H. Wu, B. Liu, J. Lu, Y. Wang, X. Wang, Y. Zhang, and L. Dong, "Popularity-driven coordinated caching in Named Data Networking," in *Proceedings of the eighth ACM/IEEE symposium on Architectures for networking and communications systems*. ACM, 2012, pp. 15–26.

[35] K. Thar, T. Z. Oo, C. Pham, S. Ullah, D. H. Lee, and C. S. Hong, "Efficient forwarding and popularity based caching for Content Centric Network," in *Information Networking (ICOIN), 2015 International Conference on*. IEEE, 2015, pp. 330–335.

[36] H. Park, H. Jang, and T. Kwon, "Popularity-based congestion control in Named Data Networking," in *Ubiquitous and Future Networks (ICUFN), 2014 Sixth International Conf on*. IEEE, 2014, pp. 166–171.

[37] H. Li, H. Nakazato, A. Detti, and N. B. Melazzi, "Popularity proportional cache size allocation policy for video delivery on CCN," in *Networks and Communications (EuCNC), 2015 European Conference on*. IEEE, 2015, pp. 434–438.

[38] K. Lei, J. Wang, and J. Yuan, "An entropy-based probabilistic forwarding strategy in Named Data Networking," in *Communications (ICC), 2015 IEEE International Conference on*. IEEE, 2015, pp. 5665–5671.

[39] J. Garcia-Reinoso, I. Vidal, D. Diez, D. Corujo, and R. L. Aguiar, "Analysis and enhancements to probabilistic caching in Content-Centric Networking," *The Computer Journal*, p. bxv010, 2015.

[40] H. Wu, J. Li, and J. Zhi, "MBP: A max-benefit probability-based caching strategy in Information Centric Networking," in *Communications (ICC), 2015 IEEE International Conference on*. IEEE, 2015, pp. 5646–5651.

[41] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for Information-Centric Networks," in *Proceedings of the second edition of the ICN workshop on Information-centric networking*. ACM, 2012, pp. 55–60.

[42] Y. Xu, Z. Wang, Y. Li, T. Lin, W. An, and S. Ci, "Minimizing bandwidth cost of CCN: A coordinated in-network caching approach," in *Computer Communication and Networks (ICCCN), 2015 24th International Conference on*. IEEE, 2015, pp. 1–7.

[43] Y. Xu, Z. Wang, Y. Li, F. Chen, T. Lin, and W. Niu, "Request routing through collaborative in-network caching for bandwidth optimization: A methodology," *Transactions on Emerging Telecommunications Technologies*, 2015.

[44] D. Perino, M. Varvello, and K. P. Puttaswamy, "ICN-RE: Redundancy elimination for Information-Centric Networking," in *Proceedings of the second edition of the ICN workshop on Information-Centric Networking*. ACM, 2012, pp. 91–96.

[45] M. Rezazad and Y. Tay, "CCNDNS: A strategy for spreading content and decoupling NDN caches," *IFIP Networking, Toulouse, France May*, pp. 20–22, 2015.

[46] X. Zhang, N. Wang, V. G. Vassilakis, and M. P. Howarth, "A distributed in-network caching scheme for P2P-like content chunk delivery," *Computer Networks*, vol. 91, pp. 577–592, 2015.

[47] V. Sourlas, P. Georgatsos, P. Flegkas, and L. Tassiulas, "Partition-based caching in Information Centric Networks," in *Computer Communications Workshops (INFOCOM WKSHPS), 2015 IEEE Conference on*. IEEE, 2015, pp. 396–401.

[48] G. Ma, Z. Chen, and K. Zhao, "A cache management strategy for content store in Content Centric Network," in *Networking and Distributed Computing (ICNDC), 2013 Fourth International Conference on*. IEEE, 2013, pp. 94–99.

[49] L. Wang, S. Bayhan, and J. Kangasharju, "Optimal chunking and partial caching in Information Centric Networks," *Computer Communications*, vol. 61, pp. 48–57, 2015.

[50] D. Kim, S.-W. Lee, Y.-B. Ko, and J.-H. Kim, "Cache capacity-aware content centric networking under flash crowds," *Journal of Network and Computer Applications*, vol. 50, pp. 101–113, 2015.

[51] S. Saha, A. Lukyanenko, and A. Ylä-Jääski, "Efficient cache availability management in Information-Centric Networks," *Computer Networks*, vol. 84, pp. 32–45, 2015.

[52] M. Mangili, F. Martignon, and S. Paraboschi, "A cache-aware mechanism to enforce confidentiality, trackability and access policy evolution in Content-Centric Networks," *Computer Networks*, vol. 76, pp. 126–145, 2015.

[53] B. Azimdoost, G. Farhadi, N. Abani, and A. Ito, "Optimal in-network cache allocation and content placement," in *Computer Communications Workshops (INFOCOM WKSHPS), 2015 IEEE Conference on*. IEEE, 2015, pp. 263–268.

[54] C. Miao, H. Zhang, H. Zhou, P. Dong, and S. Shen, "Super node routing strategy in Content-Centric Networking," *Transactions of Tianjin University*, vol. 21, pp. 122–128, 2015.

[55] H. Salah and T. Strufe, "CoMon: An architecture for coordinated caching and cache-aware routing in CCN," in *Consumer Communications and Networking Conference (CCNC), 2015 12th Annual IEEE*. IEEE, 2015, pp. 663–670.

[56] H. Wu, J. Li, and J. Zhi, "Could end system caching and cooperation replace in-network caching in CCN?" in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. ACM, 2015, pp. 101–102.

[57] J. M. Batalla, A. Beben, and Y. Chen, "Optimized decision algorithm for Information Centric Networks," *Telecommunication Systems*, pp. 1–9, 2015.

[58] C. Bernardini, "Strategies de cache basees sur la popularite pour Content Centric Networking," Ph.D. dissertation, Universite de Lorraine, France, 2015. [Online]. Available: http://www.theses.fr/2015LORR0121

[59] D. Perino and M. Varvello, "A reality check for content centric networking," in *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*. ACM, 2011, pp. 44–49.

[60] G. Rossini, D. Rossi, M. Garetto, and E. Leonardi, "Multi-terabyte and multi-gbps information centric routers," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 2014, pp. 181–189.

[61] S. Arianfar, P. Nikander, and J. Ott, "Packet-level caching for information-centric networking," in *ACM SIGCOMM, ReArch Workshop*, 2010.

[62] J. Hourcade, R. Saracco, I. Wahlster, and R. Posch, "Future Internet 2020: Visions of an industry expert group," *European Commission Information Society and Media*, vol. 5, 2009. [Online]. Available: http://www.future-internet.eu/fileadmin/documents/reports/FI_Panel_Report_v3.1_Final.pdf

[63] M. Zhang, H. Luo, and H. Zhang, "A survey of caching mechanisms in Information-Centric Networking." [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=7080842

[64] The CCNx protocol. [Online]. Available: http://www.ccnx.org/

[65] W. Wong, M. Giraldi, M. F. Magalhães, and J. Kangasharju, "Content routers: Fetching data on network path," in *Communications (ICC), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1–6.

[66] D. Kutscher, S. Eum, K. Pentikousis, I. Psaras, D. Corujo, D. Saucez, T. Schmidt, and M. Waehlisch, "ICN research challenges," *Work in progress*, 2015. [Online]. Available: https://tools.ietf.org/html/draft-irtf-icnrg-challenges-03

[67] M. Draxler and H. Karl, “Efficiency of on-path and off-path caching strategies in information centric networks,” in *Green Computing and Communications (GreenCom), 2012 IEEE International Conference on*. IEEE, 2012, pp. 581–587.

[68] X. Tang and S. T. Chanson, “Coordinated en-route web caching,” *Computers, IEEE Transactions on*, vol. 51, no. 6, pp. 595–607, 2002.

[69] H. Shen and S. Xu, “Coordinated en-route web caching in multiserver networks,” *Computers, IEEE Transactions on*, vol. 58, no. 5, pp. 605–619, 2009.

[70] M. R. Korupolu and M. Dahlin, “Coordinated placement and replacement for large-scale distributed caches,” *Knowledge and Data Engineering, IEEE Transactions on*, vol. 14, no. 6, pp. 1317–1329, 2002.

[71] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, “Summary cache: A scalable wide-area web cache sharing protocol,” *IEEE/ACM Transactions on Networking (TON)*, vol. 8, no. 3, pp. 281–293, 2000.

[72] G. M. Voelker, E. J. Anderson, T. Kimbrel, M. J. Feeley, J. S. Chase, A. R. Karlin, and H. M. Levy, “Implementing cooperative prefetching and caching in a globally-managed memory system,” in *ACM SIGMETRICS Performance Evaluation Review*, vol. 26, no. 1. ACM, 1998, pp. 33–43.

[73] B.-G. Chun, K. Chaudhuri, H. Wee, M. Barreno, C. H. Papadimitriou, and J. Kubiatowicz, “Selfish caching in distributed systems: a game-theoretic analysis,” in *Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*. ACM, 2004, pp. 21–30.

[74] N. Laoutaris, O. Telelis, V. Zissimopoulos, and I. Stavrakakis, “Distributed selfish replication,” *Parallel and Distributed Systems, IEEE Transactions on*, vol. 17, no. 12, pp. 1401–1413, 2006.

[75] M. Rabinovich, J. Chase, and S. Gadde, “Not all hits are created equal: Cooperative proxy caching over a wide-area network,” *Computer Networks and ISDN Systems*, vol. 30, no. 22, pp. 2253–2259, 1998.

[76] V. Pacifici and G. Dan, “Content-peering dynamics of autonomous caches in a Content-centric Network,” in *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2013, pp. 1079–1087.

[77] ——, “Selfish content replication on graphs,” in *Proceedings of the 23rd International Teletraffic Congress*. International Teletraffic Congress, 2011, pp. 119–126.

[78] G. Dan, “Cache-to-cache: Could ISPs cooperate to decrease peer-to-peer content distribution costs?” *Parallel and Distributed Systems, IEEE Transactions on*, vol. 22, no. 9, pp. 1469–1482, 2011.

[79] G. Rossini and D. Rossi, “Coupling caching and forwarding: Benefits, analysis, and implementation,” in *Proceedings of the 1st international conference on Information-centric networking*. ACM, 2014, pp. 127–136.

[80] D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, and K. K. Ramakrishnan, "Optimal content placement for a large-scale VoD system," in *Proceedings of the 6th International Conference*. ACM, 2010, p. 4.

[81] J. Garcia-Luna-Aceves, A. Dabirmoghaddam, and M. Mirzazad-Barijoug, "Understanding optimal caching and opportunistic caching at" the edge" of Information-Centric Networks," in *Proceedings of the 1st international conference on Information-centric networking*, 2014.

[82] J. Dai, Z. Hu, B. Li, J. Liu, and B. Li, "Collaborative hierarchical caching with dynamic request routing for massive content distribution," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 2444–2452.

[83] D. Corujo, R. L. Aguiar, I. Vidal, J. Garcia-Reinoso, and K. Pentikousis, "Research challenges towards a managed Information-Centric Network of Things," in *Networks and Communications (EuCNC), 2014 European Conference on*. IEEE, 2014, pp. 1–5.

[84] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, and D. Lewin, "Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web," in *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*. ACM, 1997, pp. 654–663.

[85] G. Barish and K. Obraczke, "World wide web caching: Trends and techniques," *IEEE Communications magazine*, vol. 38, no. 5, pp. 178–184, 2000.

[86] A. Rowstron and P. Druschel, "Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility," in *ACM SIGOPS Operating Systems Review*, vol. 35, no. 5. ACM, 2001, pp. 188–201.

[87] C. Williamson, "On filter effects in web caching hierarchies," *ACM Transactions on Internet Technology (TOIT)*, vol. 2, no. 1, pp. 47–77, 2002.

[88] M. Rabinovich and O. Spatscheck, *Web caching and replication*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002. [Online]. Available: http://www.amazon.com/Web-Caching-Replication-Michael-Rabinovich/dp/0201615703

[89] C. Kumar and J. B. Norris, "A new approach for a proxy-level web caching mechanism," *Decision Support Systems*, vol. 46, no. 1, pp. 52–60, 2008.

[90] B. Ager, F. Schneider, J. Kim, and A. Feldmann, "Revisiting cacheability in times of user generated content," in *INFOCOM IEEE Conference on Computer Communications Workshops, 2010*. IEEE, 2010, pp. 1–6.

[91] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos *et al.*, "Named Data Networking (NDN) project," *Relatório Técnico NDN-0001, Xerox Palo Alto Research Center-PARC*, 2010.

[92] A. Dabirmoghaddam, M. M. Barijough, and J. Garcia-Luna-Aceves, "Understanding optimal caching and opportunistic caching at the edge of information-centric networks," in *Proceedings of the 1st international conference on Information-centric networking*. ACM, 2014, pp. 47–56.

[93] H. Jeon, B. Lee, and H. Song, "On-path caching in Information-Centric Networking," in *Advanced Communication Technology (ICACT), 2013 15th International Conference on*. IEEE, 2013, pp. 264–267.

[94] G. Xylomenos, X. Vasilakos, C. Tsilopoulos, V. A. Siris, and G. C. Polyzos, "Caching and mobility support in a publish-subscribe internet architecture," *Communications Magazine, IEEE*, vol. 50, no. 7, pp. 52–58, 2012.

[95] Y. Wang, Z. Li, G. Tyson, S. Uhlig, and G. Xie, "Optimal cache allocation for Content-Centric Networking," in *Network Protocols (ICNP), 2013 21st IEEE International Conference on*. IEEE, 2013, pp. 1–10.

[96] L. Saino, I. Psaras, and G. Pavlou, "Hash-routing schemes for Information Centric Networking," in *Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*. ACM, 2013, pp. 27–32.

[97] Z. Li and G. Simon, "Time-shifted tv in content centric networks: The case for cooperative in-network caching," in *Communications (ICC), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1–6.

[98] H. Che, Y. Tung, and Z. Wang, "Hierarchical web caching systems: Modeling, design and experimental results," *Selected Areas in Communications, IEEE Journal on*, vol. 20, no. 7, pp. 1305–1314, 2002.

[99] The Binomial Distribution. [Online]. Available: http://www.hamilton.ie/ollie/EE304/Binom.pdf

[100] L. Ramaswamy and L. Liu, "An expiration age-based document placement scheme for cooperative web caching," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 16, no. 5, pp. 585–600, 2004.

[101] E. J. Rosensweig and J. Kurose, "Breadcrumbs: Efficient, best-effort content location in cache networks," in *INFOCOM 2009, IEEE*. IEEE, 2009, pp. 2631–2635.

[102] Y. Li, T. Lin, H. Tang, and P. Sun, "A chunk caching location and searching scheme in Content Centric Networking," in *Communications (ICC), 2012 IEEE International Conference on*. IEEE, 2012, pp. 2655–2659.

[103] G. Zhang, B. Tang, X. Wang, and Y. Wu, "An optimal cache placement strategy based on content popularity in Content Centric Network," *Journal of Information & Computational Science*, vol. 11, no. 8, pp. 2759–2769, 2014.

[104] S. Eum, K. Nakauchi, Y. Shoji, and N. Nishinaga, "CATT: Cache aware target identification for ICN," *Communications Magazine, IEEE*, vol. 50, no. 12, pp. 60–67, 2012.

[105] S. Eum, K. Nakauchi, M. Murata, Y. Shoji, and N. Nishinaga, "CATT: Potential based routing with content caching for ICN," in *Proceedings of the 2nd ACM SIGCOMM workshop on Information-centric networking*. ACM, 2012, pp. 49–54.

[106] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache less for more in Information-Centric Networks (Extended Version)," *Computer Communications*, vol. 36, no. 7, pp. 758–770, 2013.

[107] J. Sung, J.-K. K. Rhee, and S. Jung, "Lightweight caching strategy for wireless content delivery networks," *IEICE Communications Express*, vol. 3, no. 4, pp. 150–155, 2014.

[108] J. Wang, J. Ren, K. Lu, J. Wang, S. Liu, and C. Westphal, "An optimal cache management framework for Information Centric Networks with network coding," in *Networking Conference, 2014 IFIP*. IEEE, 2014, pp. 1–9.

[109] M. Bilal and S.-G. Kang, "Time Aware Least Recent Used (TLRU) cache management policy in ICN," in *Advanced Communication Technology (ICACT), 2014 16th International Conference on*. IEEE, 2014, pp. 528–532.

[110] K. Cho, M. Lee, K. Park, T. T. Kwon, Y. Choi, and S. Pack, "WAVE: Popularity-based and collaborative in-network caching for Content-oriented Networks," in *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*. IEEE, 2012, pp. 316–321.

[111] Z. Ming, M. Xu, and D. Wang, "Age-based cooperative caching in Information-Centric Networking," in *Computer Communication and Networks (ICCCN), 2014 23rd International Conference on*. IEEE, 2014, pp. 1–8.

[112] X. Hu and J. Gong, "Distributed in-network cooperative caching," in *Cloud Computing and Intelligent Systems (CCIS), 2012 IEEE 2nd International Conference on*, vol. 2. IEEE, 2012, pp. 735–740.

[113] W. Tianming, C. Le, Y. Boyang, and P. Jianping, "MPCS: A mobility/popularity-based caching strategy for Information Centric Networks," in *Global Communications Conference (GLOBECOM), 2014 IEEE Conference on*. IEEE, 2014, pp. 4629–2634.

[114] M. Ong, M. Chen, T. Taleb, X. Wang, and V. Leung, "FGPC: Fine-grained popularity-based caching design for Content Centric Networking," in *Proceedings of the 17th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*. ACM, 2014, pp. 295–302.

[115] G. Carofiglio, M. Gallo, L. Muscariello, and D. Perino, "Modeling data transfer in Content-Centric Networking," in *Proceedings of the 23rd international teletraffic congress*. International Teletraffic Congress, 2011, pp. 111–118.

[116] I. Psaras, R. G. Clegg, R. Landa, W. K. Chai, and G. Pavlou, "Modelling and evaluation of CCN-caching trees," in *NETWORKING 2011*. Springer, 2011, pp. 78–91.

[117] L. Muscariello, G. Carofiglio, and M. Gallo, "Bandwidth and storage sharing performance in Information Centric Networking," in *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*. ACM, 2011, pp. 26–31.

[118] G. Carofiglio, V. Gehlen, and D. Perino, "Experimental evaluation of memory management in Content-Centric Networking," in *Communications (ICC), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1–6.

[119] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker, "Less pain, most of the gain: Incrementally deployable ICN," in *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4. ACM, 2013, pp. 147–158.

[120] Y. Thomas and G. Xylomenos, "Towards improving the efficiency of ICN packet-caches," in *Heterogeneous Networking for Quality, Reliability, Security and Robustness (QShine), 2014 10th International Conference on*. IEEE, 2014, pp. 186–187.

[121] A. Gharaibeh, A. Khreishah, I. Khalil, and J. Wu, "Asymptotically-optimal incentive-based en-route caching scheme," *arXiv preprint arXiv*, vol. 1404, 2014. [Online]. Available: http://arxiv.org/pdf/1404.4639.pdf

[122] J. M. Wang, J. Zhang, and B. Bensaou, "Intra-AS cooperative caching for Content-Centric Networks," in *Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*. ACM, 2013, pp. 61–66.

[123] M. Badov, A. Seetharam, J. Kurose, V. Firoiu, and S. Nanda, "Congestion-aware caching and search in Information-Centric Networks," in *Proceedings of the 1st international conference on Information-centric networking*. ACM, 2014, pp. 37–46.

[124] W.-X. Liu, S.-Z. Yu, Y. Gao, and W.-T. Wu, "Caching efficiency of Information-Centric Networking," *IET networks*, vol. 2, no. 2, pp. 53–62, 2013.

[125] T.-M. Pham, M. Minoux, S. Fdida, M. Pilarski *et al.*, "Optimization of content caching in Content-Centric Network," 2014. [Online]. Available: http://hal.upmc.fr/hal-01016470/file/caching_Paper_ICN_Final.pdf

[126] N. B. Melazzi, A. Detti, M. Arumaithurai, and K. Ramakrishnan, "Internames: A name-to-name principle for the Future Internet," in *Heterogeneous Networking for Quality, Reliability, Security and Robustness (QShine), 2014 10th International Conference on*. IEEE, 2014, pp. 146–151.

[127] S. Wang, J. Bi, J. Wu, Z. Li, W. Zhang, and X. Yang, "Could in-network caching benefit Information Centric Networking?" in *Proceedings of the 7th Asian Internet Engineering Conference*. ACM, 2011, pp. 112–115.

[128] L. Zhang, J. Zhao, and Z. Shi, "LF: A caching strategy for named data mobile ad hoc networks," in *Proceedings of the 4th International Conference on Computer Engineering and Networks*. Springer, 2015, pp. 279–290.

[129] V. G. Vassilakis, M. F. Al-Naday, M. J. Reed, B. Alzahrani, K. Yang, I. D. Moscholios, M. D. Logothetis *et al.*, “A cache-aware routing scheme for Information Centric Networks,” in *Communication Systems, Networks & Digital Signal Processing (CSNDSP), 2014 9th International Symposium on*. IEEE, 2014, pp. 721–726.

[130] J. Garcia-Luna-Aceves, “A fault-tolerant forwarding strategy for interest-based information centric networks,” in *IFIP Networking Conference (IFIP Networking), 2015*. IEEE, 2015, pp. 1–9.

[131] Y. Li, H. Xie, Y. Wen, C.-Y. Chow, and Z.-L. Zhang, “How much to coordinate? optimizing in-network caching in Content-Centric Networks,” *Network and Service Management, IEEE Transactions on*, vol. 12, no. 3, pp. 420–434, 2015.

[132] L. T. Blessing and A. Chakrabarti, *DRM, a Design Research Methodology*. Springer, 2009. [Online]. Available: http://www.springer.com/us/book/9781848825864

[133] A. Habbal, “TCP-Sintok: Transmission Control Protocol with loss detection and contention avoidance mechanisms for ad hoc networks,” Ph.D. dissertation, School of Computing, Universiti Utara Malaysia, 2014. [Online]. Available: http://etd.uum.edu.my/4442/13/s92256_abstract.pdf

[134] Methodology for Computer Science. [Online]. Available: http://www.uio.no/studier/emner/matnat/ifi/INF9970/h09/undervisningsmateriale/ResearchMethods-CS.pdf

[135] S.-J. Chen and C.-L. Hwang, *Fuzzy multiple attribute decision making methods*. Springer, 1992. [Online]. Available: http://link.springer.com/book/10.1007%2F978-3-642-46768-4

[136] P. TalebiFard and V. C. Leung, “A data fusion approach to context-aware service delivery in heterogeneous network environments,” *Procedia Computer Science*, vol. 5, pp. 312–319, 2011.

[137] P. Talebi Fard, “An Information Centric Networking approach to context-aware dissemination of services and information,” Ph.D. dissertation, University of British Columbia, 2014. [Online]. Available: https://open.library.ubc.ca/cIRcle/collections/ubctheses/24/items/1.0167484

[138] M. Guizani, A. Rayes, B. Khan, and A. Al-Fuqaha, *Network modeling and simulation: A practical perspective*. John Wiley & Sons, 2010. [Online]. Available: http://as.wiley.com/WileyCDA/WileyTitle/productCd-0470035870.html

[139] *IEEE Standards Board, IEEE recommended practice for distributed interactive simulation 2013: Verification, validation, and accreditation*, IEEE Standard 1278.4-1997, January 2010. [Online]. Available: https://standards.ieee.org/findstds/standard/1730.1-2013.html

[140] D. Thomas, A. Joiner, W. Lin, M. Lowry, and T. Pressburger, "The unique aspects of simulation verification and validation," in *Aerospace Conference, 2010 IEEE.* IEEE, 2010, pp. 1–7. [Online]. Available: http://ti.arc.nasa.gov/m/profile/ttp/ieee_aero_2010.pdf

[141] R. G. Sargent, "Validation and verification of simulation models," in *Simulation Conference, 2004. Proceedings of the 2004 Winter*, vol. 1. IEEE, 2004. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5679166&tag=1

[142] C. Bernardini. SocialCCNSim. [Online]. Available: https://github.com/mesarpe/socialccnsim

[143] OPNET Modeler. [Online]. Available: http://www.riverbed.com/my/products/steelcentral/opnet.html?redirect=opnet

[144] R. Jain, "The art of computer system performance analysis: Techniques for experimental design, measurement, simulation and modeling," *New York: John Willey*, 1991. [Online]. Available: http://as.wiley.com/WileyCDA/WileyTitle/productCd-0471503363.html

[145] S. Hassan, "Simulation-based performance evaluation of TCP-friendly protocols for supporting multimedia applications in the Internet," Ph.D. dissertation, University of Leeds (School of Computing), 2002. [Online]. Available: http://myto.upm.edu.my/myTO/myto/15/paparthesis/161367.html

[146] O. M. Al-Momani, "Dynamic redundancy forward error correction mechanism for the enhancement of Internet-based video streaming," Ph.D. dissertation, School of Computing, Universiti Utara Malaysia, 2010. [Online]. Available: http://etd.uum.edu.my/2523/

[147] O. Ghazali, "Scalable and smooth TCP-friendly receiver-based layered multicast protocol," Ph.D. dissertation, School of Computing, Universiti Utara Malaysia, 2008. [Online]. Available: http://etd.uum.edu.my/1291/2/Osman_Ghazali.pdf

[148] C. Williamson, "Internet traffic measurement," *Internet Computing, IEEE*, vol. 5, no. 6, pp. 70–74, 2001.

[149] B. L. Ong, "A hybrid mechanism for SIP over IPv6 macromobility and micromobility management protocols," Ph.D. dissertation, School of Computing, Universiti Utara Malaysia, 2008. [Online]. Available: http://etd.uum.edu.my/1256/

[150] The Network Simulator - ns-2. [Online]. Available: http://www.isi.edu/nsnam/ns/

[151] A discrete-event network simulator - ns-3. [Online]. Available: https://www.nsnam.org/

[152] J. Jaseem, "Performance comparison between ad hoc on demand distance vector and dynamic source routing protocols with security encryption using OPNET," 2012. [Online]. Available: http://scholarworks.rit.edu/cgi/viewcontent.cgi?article=1609&context=theses

[153] CCNx Simulator. [Online]. Available: http://www.ccnx.org/

[154] CCNPLS. CCN Packet Level Simulator. [Online]. Available: https://code.google.com/p/ccnpl-sim/

[155] R. Chiocchetti, D. Rossi, and G. Rossini, "ccnSim: An highly scalable CCN simulator," in *Communications (ICC), 2013 IEEE International Conference on*. IEEE, 2013, pp. 2309–2314.

[156] A. Afanasyev, I. Moiseenko, L. Zhang *et al.*, "ndnSIM: NDN simulator for NS-3," *University of California, Los Angeles, Tech. Rep*, 2012. [Online]. Available: http://named-data.net/wp-content/uploads/TRndnsim.pdf

[157] ICN Simulator: A Simulator based on the Blackadder paltform. [Online]. Available: http://privatewww.essex.ac.uk/~nvasta/ICNSim.htm

[158] A. Varga and R. Hornig, "An overview of the OMNeT++ simulation environment," in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, p. 60.

[159] L. Saino, I. Psaras, and G. Pavlou, "Icarus: A caching simulator for Information Centric Networking (ICN)," in *Proceedings of the 7th International ICST Conference on Simulation Tools and Techniques*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2014, pp. 66–75.

[160] T. R. Henderson, S. Roy, S. Floyd, and G. F. Riley, "ns-3 project goals," in *Proceeding from the 2006 workshop on ns-2: the IP network simulator*. ACM, 2006, p. 13.

[161] PAL Project. [Online]. Available: http://palproject.org.uk/

[162] C. Bernardini, T. Silverston, and O. Festor, "SONETOR: A social network traffic generator," in *Communications (ICC), 2014 IEEE International Conference on*. IEEE, 2014, pp. 3734–3739.

[163] J. Ren, W. Qi, C. Westphal, J. Wang, K. Lu, S. Liu, and S. Wang, "MAGIC: A distributed MAx-Gain In-network Caching strategy in information-centric networks," in *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*. IEEE, 2014, pp. 470–475.

[164] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache less for more in Information-Centric Networks," in *Volume 7289 of the series Lecture Notes in Computer Science (NETWORKING 2012)*. Springer, 2012, pp. 27–40. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-642-30045-5_3

[165] J. Leskovec and A. Krevl, “SNAP Datasets: Stanford large network dataset collection,” Jun. 2014. [Online]. Available: http://snap.stanford.edu/data

[166] J. Leskovec and J. J. Mcauley, “Learning to discover social circles in ego networks,” in *Advances in neural information processing systems*, 2012, pp. 539–547. [Online]. Available: http://cs.stanford.edu/people/jure/pubs/circles-nips12.pdf

[167] Stanford Network Analysis Project (SNAP). [Online]. Available: http://snap.stanford.edu/data/egonets-Facebook.html

[168] LastFM: A UK-based music website found in 2002. [Online]. Available: http://www.last.fm/

[169] I. Cantador, P. Brusilovsky, and T. Kuflik, “Second workshop on information heterogeneity and fusion in recommender systems (hetrec2011).” in *RecSys*, 2011, pp. 387–388. [Online]. Available: http://ir.ii.uam.es/reshet/pub/recsys11d.pdf

[170] J. Guo, W. Xiang, and S. Wang, “Reinforce networking theory with OPNET simulation,” *Journal of Information Technology Education: Research*, vol. 6, no. 1, pp. 215–226, 2007.

[171] A. C. Onwutalobi, “TCP congestion control,” Department of Computer Science, University of Helsinki, Tech. Rep., 2005. [Online]. Available: https://www.academia.edu/18593436/TCP_Congestion_Control

[172] M. Hassan and R. Jain, *High performance TCP/IP networking*. Prentice Hall, 2003. [Online]. Available: https://www.cse.wustl.edu/~jain/books/ftp/tcp_fm.pdf

[173] I. Ud, Din, S. Hassan, and A. Habbal, “SocialCCNSim: A simulator for caching strategies in Information Centric Networking,” *Advanced Science Letters*, vol. 21, no. 11, pp. 3507–3511, 2015.

[174] D. Rossi and G. Rossini, “Caching performance of Content Centric Networks under multi-path routing (and more),” *Relatório técnico, Telecom ParisTech*, 2011. [Online]. Available: http://netlab.pkusz.edu.cn/wordpress/wp-content/uploads/2011/10/

[175] C. Fricker, P. Robert, J. Roberts, and N. Sbihi, “Impact of traffic mix on caching performance in a Content-Centric Network,” in *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*. IEEE, 2012, pp. 310–315.

[176] C. Dannewitz, “Netinf: An Information-Centric design for the Future Internet,” in *Proc. 3rd GI/ITG KuVS Workshop on The Future Internet*, 2009. [Online]. Available: http://www-old.cs.uni-paderborn.de/fileadmin/Informatik/AG-Karl/Publications/KuVS-NetInf-Dannewitz.pdf

[177] W. K. Wong, L. Wang, and J. Kangasharju, “Neighborhood search and admission control in cooperative caching networks,” in *Global Communications Conference (GLOBECOM), 2012 IEEE*. IEEE, 2012, pp. 2852–2858.

[178] L. Wang, S. Bayhan, and J. Kangasharju, "Effects of cooperation policy and network topology on performance of in-network caching," *Communications Letters, IEEE*, vol. 18, no. 4, pp. 680–683, 2014.

[179] Y. Xu, Y. Li, T. Lin, Z. Wang, W. Niu, H. Tang, and S. Ci, "A novel cache size optimization scheme based on manifold learning in Content Centric Networking," *Journal of Network and Computer Applications*, vol. 37, pp. 273–281, 2014.

[180] J. Ardelius, B. Gronvall, L. Westberg, and A. Arvidsson, "On the effects of caching in access aggregation networks," in *Proceedings of the second edition of the ICN workshop on Information-centric networking*. ACM, 2012, pp. 67–72.

[181] A. Ravi, P. Ramanathan, and K. M. Sivalingam, "Integrated network coding and caching in information-centric networks: revisiting pervasive caching in the ICN framework," *Photonic Network Communications*, vol. 30, no. 3, pp. 416–427, 2015.

[182] A. Anand, C. Muthukrishnan, A. Akella, and R. Ramjee, "Redundancy in network traffic: findings and implications," *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 1, pp. 37–48, 2009.

[183] A. Anand, V. Sekar, and A. Akella, "Smartre: An architecture for coordinated network-wide redundancy elimination," in *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4. ACM, 2009, pp. 87–98.

[184] A. Anand, A. Gupta, A. Akella, S. Seshan, and S. Shenker, "Packet caches on routers: the implications of universal redundant traffic elimination," in *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4. ACM, 2008, pp. 219–230.

[185] E. Zohar, I. Cidon, and O. O. Mokryn, "The power of prediction: Cloud bandwidth and cost reduction," in *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4. ACM, 2011, pp. 86–97.

[186] N. T. Spring and D. Wetherall, "A protocol-independent technique for eliminating redundant network traffic," *ACM SIGCOMM Computer Communication Review*, vol. 30, no. 4, pp. 87–95, 2000.

[187] S. Ullah and C. S. Hong, "Probabilistic in-networking chunk marking and caching for Information-Centric Networks," *2013 Korea Information Science Society 40th Annual General Meeting and Fall Conference*, pp. 964–966, 2013.

[188] L. Wang, W. Wong, and J. Kangasharju, "In-network caching vs. redundancy elimination," *CoRR*, vol. abs/1311.7421, 2013. [Online]. Available: http://arxiv.org/abs/1311.7421

[189] R. S. Antunes, M. B. Lehmann, R. B. Mansilha, C. Esteve Rothenberg, L. P. Gaspary, and M. P. Barcellos, "Ccnrel: Leveraging relations among objects to improve the performance of CCN," in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*. IEEE, 2015, pp. 199–206.

[190] Y. Wang, K. Lee, B. Venkataraman, R. L. Shamanna, I. Rhee, and S. Yang, “Advertising cached contents in the control plane: Necessity and feasibility,” in *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*. IEEE, 2012, pp. 286–291.

[191] Q. N. Nguyen, M. Arifuzzaman, T. Miyamoto, and S. Takuro, “An optimal information centric networking model for the future green network,” in *Autonomous Decentralized Systems (ISADS), 2015 IEEE Twelfth International Symposium on*. IEEE, 2015, pp. 272–277.

[192] J. Wang, “A survey of web caching schemes for the Internet,” *ACM SIGCOMM Computer Communication Review*, vol. 29, no. 5, pp. 36–46, 1999.

[193] D. Wessels and K. Claffy, “RFC 2186: Internet Cache Protocol (ICP), version 2, September 1997,” *Status: INFORMATIONAL*. [Online]. Available: https://tools.ietf.org/html/rfc2186

[194] ——, “RFC 2187: Application of Internet Cache Protocol (ICP), version 2, September 1997,” *Status: INFORMATIONAL*, vol. 233, p. 234. [Online]. Available: https://tools.ietf.org/html/rfc2187

[195] S. Michel, K. Nguyen, A. Rosenstein, L. Zhang, S. Floyd, and V. Jacobson, “Adaptive web caching: towards a new global caching architecture,” *Computer Networks and ISDN systems*, vol. 30, no. 22, pp. 2169–2177, 1998.

[196] E. Nygren, R. K. Sitaraman, and J. Sun, “The akamai network: a platform for high-performance internet applications,” *ACM SIGOPS Operating Systems Review*, vol. 44, no. 3, pp. 2–19, 2010.

[197] T. Krenc, O. Hohlfeld, and A. Feldmann, “An Internet census taken by an illegal botnet: A qualitative assessment of published measurements,” *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 103–111, 2014.

[198] L. A. Zadeh, “Fuzzy sets,” *Information and control*, vol. 8, no. 3, pp. 338–353, 1965. [Online]. Available: http://www.cs.berkeley.edu/~zadeh/papers/Fuzzy%20Sets-Information%20and%20Control-1965.pdf

[199] K. R. MacCrimmon, “Decision making among multiple-attribute alternatives: A survey and consolidated approach,” DTIC Document, Tech. Rep., 1968. [Online]. Available: https://www.rand.org/content/dam/rand/pubs/research_memoranda/2009/RM4823.pdf

[200] C.-L. Hwang and K. Yoon, *Multiple attribute decision making, methods and applications: A state-of-the-art survey*. Springer, 1981. [Online]. Available: http://link.springer.com/book/10.1007%2F978-3-642-48318-9

[201] H. Kwakernaak, “An algorithm for rating multiple-aspect alternatives using fuzzy sets,” *Automatica*, vol. 15, no. 5, pp. 615–616, 1979. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0005109879900104

[202] A. Easton, “One-of-a-kind decisions involving weighted multiple objectives and disparate alternatives,” *Multiple criteria decision making (1st ed.). University of South Carolina Press: Columbia*, pp. 657–667, 1973.

[203] K. Katsaros, G. Xylomenos, and G. C. Polyzos, “MultiCache: An overlay architecture for Information-Centric Networking,” *Computer Networks*, vol. 55, no. 4, pp. 936–947, 2011.

[204] M. Diallo, S. Fdida, V. Sourlas, P. Flegkas, and L. Tassiulas, “Leveraging caching for Internet-scale content-based publish/subscribe networks,” in *Communications (ICC), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1–5.

[205] H. Yao, C. Fang, C. Qiu, C. Zhao, and Y. Liu, “A novel energy efficiency algorithm in green mobile networks with cache,” *EURASIP Journal on Wireless Communications and Networking*, vol. 2015, no. 1, pp. 1–9, 2015.

[206] E. Triantaphyllou and C.-T. Lin, “Development and evaluation of five fuzzy multiattribute decision-making methods,” *international Journal of Approximate reasoning*, vol. 14, no. 4, pp. 281–310, 1996.

[207] L. Saino, “On the Design of Efficient Caching Systems,” Ph.D. dissertation, University College London, UK, 2015. [Online]. Available: http://discovery.ucl.ac.uk/1473436/1/thesis-final.pdf

[208] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, “Web caching and zipf-like distributions: Evidence and implications,” in *INFOCOM’99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1. IEEE, 1999, pp. 126–134.

[209] M. Gallo, B. Kauffmann, L. Muscariello, A. Simonian, and C. Tanguy, “Performance evaluation of the random replacement policy for networks of caches,” in *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 1. ACM, 2012, pp. 395–396.

[210] M. Won and R. Stoleru, “A low-stretch-guaranteed and lightweight geographic routing protocol for large-scale wireless sensor networks,” *ACM Transactions on Sensor Networks (TOSN)*, vol. 11, no. 1, p. 18, 2014.

[211] L. Wang, A. Hoque, C. Yi, A. Alyyan, and B. Zhang, “OSPFN: An OSPF based routing protocol for Named Data Networking,” *University of Memphis and University of Arizona, Tech. Rep*, 2012. [Online]. Available: http://new.named-data.net/wp-content/uploads/TROSPFN.pdf

[212] G. Carofiglio, M. Gallo, L. Muscariello, and M. Papali, “Multipath congestion control in Content-Centric Networks,” in *Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on*. IEEE, 2013, pp. 363–368.

[213] A. Ghodsi, T. Koponen, J. Rajahalme, P. Sarolahti, and S. Shenker, “Naming in content-oriented architectures,” in *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*. ACM, 2011, pp. 1–6.

[214] L. A. Adamic and B. A. Huberman, “The web’s hidden order,” *Communications of the ACM*, vol. 44, no. 9, pp. 55–60, 2001.

[215] N. B. Ellison *et al.*, “Social network sites: Definition, history, and scholarship,” *Journal of Computer-Mediated Communication*, vol. 13, no. 1, pp. 210–230, 2007.

[216] L. A. Adamic and B. A. Huberman, “Zipf’s law and the Internet,” *Glottometrics*, vol. 3, no. 1, pp. 143–150, 2002. [Online]. Available: http://www.hpl.hp.com/research/idl/papers/ranking/adamicglottometrics.pdf

[217] C. Bernardini, T. Silverston, and O. Festor, “Socially-aware caching strategy for Content Centric Networking,” in *Networking Conference, 2014 IFIP*. IEEE, 2014, pp. 1–9.

[218] A. L. Van Den Wollenberg, “Redundancy analysis an alternative for canonical correlation analysis,” *Psychometrika*, vol. 42, no. 2, pp. 207–219, 1977. [Online]. Available: http://link.springer.com/article/10.1007/BF02294050

[219] J. Xu and X.-C. Zhu, “A load-balancing and energy-aware routing protocol for MANET accessing Internet,” in *2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing*. IEEE, 2009, pp. 571–574.

[220] D. Rossi, G. Rossini *et al.*, “On sizing CCN content stores by exploiting topological information.” in *INFOCOM Workshops*, 2012, pp. 280–285. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6193506&tag=1

[221] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox, “Information-Centric Networking: Seeing the forest for the trees,” in *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*. ACM, 2011, pp. 1–6.

[222] A. Wolman, M. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. M. Levy, “On the scale and performance of cooperative web proxy caching,” in *ACM SIGOPS Operating Systems Review*, vol. 33, no. 5. ACM, 1999, pp. 16–31.

[223] S. Ihm and V. S. Pai, “Towards understanding modern web traffic,” in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. ACM, 2011, pp. 295–312.

[224] D. Kutscher, S. Eum, K. Pentikousis, I. Psaras, D. Corujo, D. Saucez, T. Schmidt, and M. Waehlisch, “ICN research challenges,” *Work in progress*, 2014. [Online]. Available: https://tools.ietf.org/html/draft-kutscher-icnrg-challenges-02