WASEDA UNIVERSITY

MASTER'S THESIS

LCD-Based Probabilistic Caching for Information Centric Networking

Author:
Abdul Milad SIDDIQUE

Supervisor:
NAKAZATO Hidenori

Student ID: 5115FG21-3

A thesis submitted in fulfillment of the requirements for the degree of Master of Engineering

in the

Department of Computer Science and Communication Engineering

July 27, 2017

Waseda University

Abstract

Faculty of Science and Engineering

Department of Computer Science and Communication Engineering

Master of Engineering

LCD-Based Probabilistic Caching for Information Centric Networking

by Abdul Milad SIDDIQUE

By the rapid increase of the content-oriented applications, storing and distributing contents become more valuable. As a future of the Internet architecture, Information-Centric Networking (ICN) has a novel feature called *in-network caching*, to cache the content in ICN routers. It avoids delivery of the same content many times in the same path and reduces user-perceived delay, server load, waste of bandwidth, etc. Recently many algorithms were proposed for the cache placement strategy. One of the traditional cache placement policy is Leave Copy Everywhere (LCE). LCE caches a copy of data passing through an ICN router. LCE is not efficient because of cache redundancy. In this work, we propose a new algorithm called *LCD-Based Probabilistic Caching* (LBPC). When a data packet is delivered back to the requester, the first ICN router located closer to the server caches the content with the highest probability, and the cache probability at each router decreases along the path. When multiple requests are sent for the same content, the copy of the content is pushed to the downstream routers hop by hop. Our simulation work shows LBPC has better performance than *In-Network Probabilistic Caching (ProbCache) LCE*, and *Leave Copy Down (LCD)*.

Acknowledgements

My foremost gratitude goes to my supervisor, Prof. Nakazato Hidenori, for the continuous support of my Master's research with his exceptional patience and wisdom. It is true that he always motivate me to work hard, his advice on both research as well as my career have been invaluable.

My gratitude extends to my lab members, friends, and colleagues for their warm support during my stay in Japan. This is the time to thank from Japan International Cooperation Agency (JICA) for their financial support and the government of Afghanistan for providing me an opportunity to join the PEACE Project for study of my Master's program at Waseda University.

Finally, I thanks my beloved family for their love and encouragement through my years of study, specially for my parents who raised me and supported me in all my pursuits.

Abdul Milad Siddique

Waseda University

July 2017

Contents

List of Figures

List of Tables

List of Abbreviations

ICN	**I**nformation **C**entric **N**etworking
NDN	**N**amed **D**ata **N**etworking
TCP/IP	**T**ransmission **C**ontrol **P**rotocol / **I**nternet **P**rotocol
ProbCache	**P**robabilistic **I**n -**N**etwork **C**aching
LCE	**L**eave **C**opy **E**vrywhere
LCD	**L**eave **C**opy **D**own
MCD	**M**ove **C**opy **D**own
CS	**C**ontent **S**tore
PIT	**P**ending **I**nterest **T**able
FIB	**F**orwarding **I**nformation **B**ase
DoS	**D**enial of **S**ervice
DDos	**D**istributed **D**enial **S**ervice
NAT	**N**etwork **A**ddress **T**ranslation
CDN	**C**ontent **D**elivery **N**etworks
LRU	**L**east **R**ecently **U**sed
LFU	**L**east **F**requently **U**sed
FIFO	**F**irst-in **F**irst-out

List of Symbols

C_i	Cache capacity of the (R_i)
R_d	The router with number of hops from producer side
c	Hop distance from the consumer
d	Hop distance from the server

Published Work

[1] S. A. Milad, "BS-5-17 Probabilistic Caching for Information Centric Networking BS-5-17," pp. 84?85, 2016.

[2] Siddique Abdul Milad and Hidenori Nakazato, LCD-Based Probabilistic Caching for Information Centric Networking, Technical Report of IEICE, CS2017, July 2017.

[3] Siddique Abdul Milad and Hidenori Nakazato, LCD-Based Probabilistic Edge Caching for Information Centric Networking, International Conference of IEICE, CS2017, September 2017.

Chapter 1

INTRODUCTION

1.1 Background and Motivation

With the advent of Information Centric Networking (ICN) [1]. The network evolve from a simple point to point communication to a content centric distribution. In the current Internet architecture a user has to specify a host in order to retrieve the content but in ICN the communication is performed by specifying the name of the requested content. The new data network architecture, ICN, solves many issues confronting the current Internet architecture such as distribution of popular content , congestion, security etc. There are many proposal of ICN. One of the main idea contained in ICN is in-network caching.

in-network caching has aroused research interest in these years. There are many algorithms proposed by researchers in terms of caching the content in ICN routers and the principle task of in-network caching research is designing an effective caching strategies to improve the system performance.

In this thesis we propose a new algorithm to cache the most popular contents at the network edge closer to the user and the less popular content to be cached to the intermediate routers. By caching the popular contents at the edge network, a lots of Interest packet can be satisfied from the nearest router, which cause shorter delay for obtaining the content, lower server load, and increase of available network bandwidth.

Our proposed algorithms can achieve a higher cache hit ratio, lower server

load and shorter average hop count compared with the current proposed algorithms which will be discuss in chapter. 4 and 5.

1.2 Thesis Organization

This thesis is organized as follows: In Chapter 2, we give background knowledge of NDN, one of the most quoted ICN implementation. Chapter 3 introduces basic idea and design of our proposed caching algorithm, LCD-Based Probabilistic Caching (LBPC) and we compare our proposed algorithm with the existing proposed algorithms such as ProbCache, LCE, and LCD [11][12]. Chapter 4 further develops the idea of LBPC by assigning the higher probability to the edge network closer to the consumer and producer and compare the the developed algorithm with LBPC, ProbCache, LCE, and LCD.

Chapter 2

BACKGROUND

2.1 Overview of Named Data Networking

In this section, we give a brief description of NDN architecture [2]. Today's internet architecture uses IP addresses for communication and the conversation is between two machines. IP packets contains two identifiers (addresses) one for source and the other one for destination host. Unlike today's internet architecture, NDN communicate by using content name. Fig. 2.1 compares the Internet and NDN protocol stack. NDN differs from the Internet in terms of strategy and security which are shown as a new layer in its protocol stack. NDN can take maximum advantage of multiple connections such as: Ethernet, 3G, Bluetooth and 802.11. In NDN content secures itself, instead of the connection over which it travels.

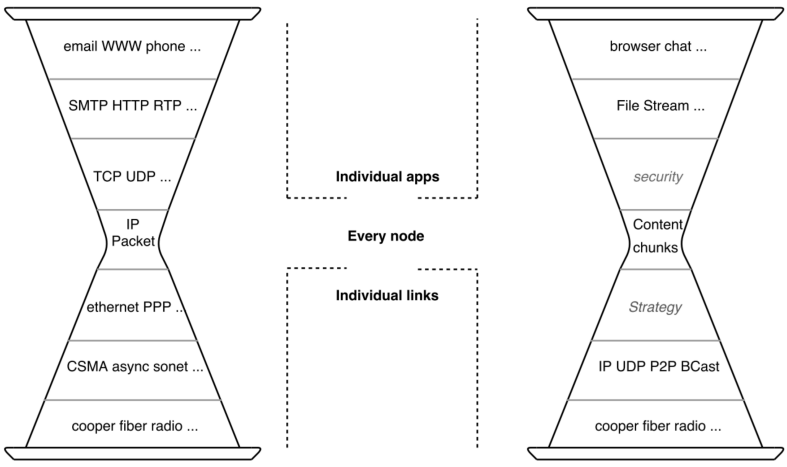

FIGURE 2.1: NDN and IP protocol stack.

NDN can communicate by exchanging two types of packets: Interest packet and Data packet as shown in Fig. 2.2 [2]. Both Interest and Data packets carry the name of the content. ICN routers forward the Interest packet to the producer (original server) by using the name. When the Interest packet meets the node which has the requested content, the node returns the Data packet which contains both the name and the content as well as a signature by the producer's key. The Data packet is forwarded on the reverse path of the request to get back to the requester.

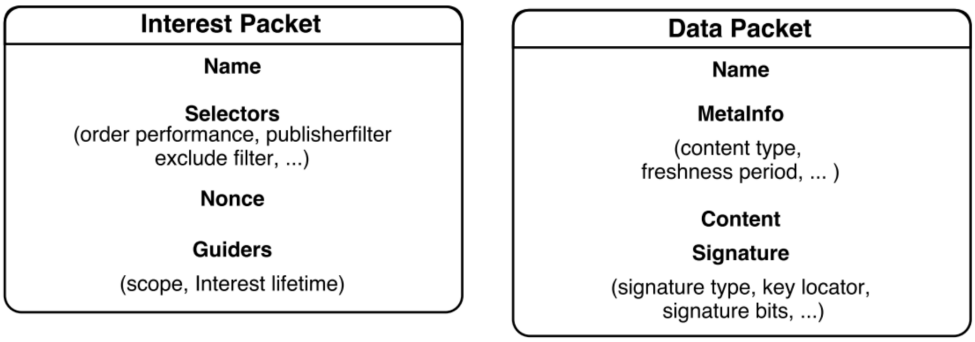

FIGURE 2.2: NDN packet types

2.2 NDN Node Data Process Model

NDN routers have three tables: A Forwarding Information Base (FIB), Pending Interest Table (PIT) and Content Store (CS). FIB defines where to forward the Interest packet based on content name. PIT is responsible for storing information of unsatisfied Interest packets forwarded by this router. The information includes the names of the contents requested by the pending Interested packets, and the incoming and outgoing interfaces of the Interest packets. CS holds the cached Data packets to satisfy the future request. The communication in NDN is based on the name of the content and neither Interest nor Data packet carries host information (IP address).

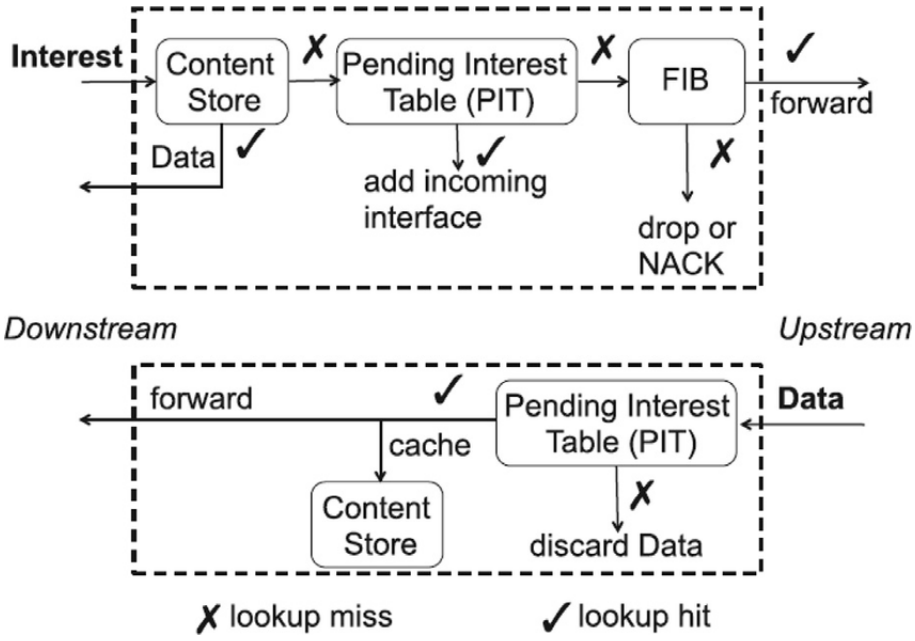

FIGURE 2.3: NDN forwarding process

On arrival of an Interest packet, an NDN router first checks its CS for the availability of the requested content. If the content does not exist, it checks its PIT. If no entry for the content name of the Interest packet exists in the PIT, the NDN router creates an entry with the content name and records the incoming interface of the Interest packet in the PIT and forward the Interest packet according to the information in the FIB. If the content name of the Interest packet already exists in PIT, it means an Interest packet is pending for the same content name. The incoming interface of the Interest packet is appended to the corresponding PIT entry and the Interest packet is discarded in this case. If an NDN router receive multiple packets for the same content name, the router forwards only the first Interest packet to the upstream router. When a Data packet arrives at an NDN router, it caches the content in the Content Store depending on the caching policy of the router. Then it checks the PIT entry if there is a pending Interest for the same content name or not. If the content name exists, it forwards the Data packet to all incoming interface stored in the PIT entry for the same content name, and erase the PIT entry. If not, the Data packet is discarded. The Data packet is always delivered through the reverse path of the Interest packet. The flow chart

of basic NDN operation shows in Fig. 2.4.

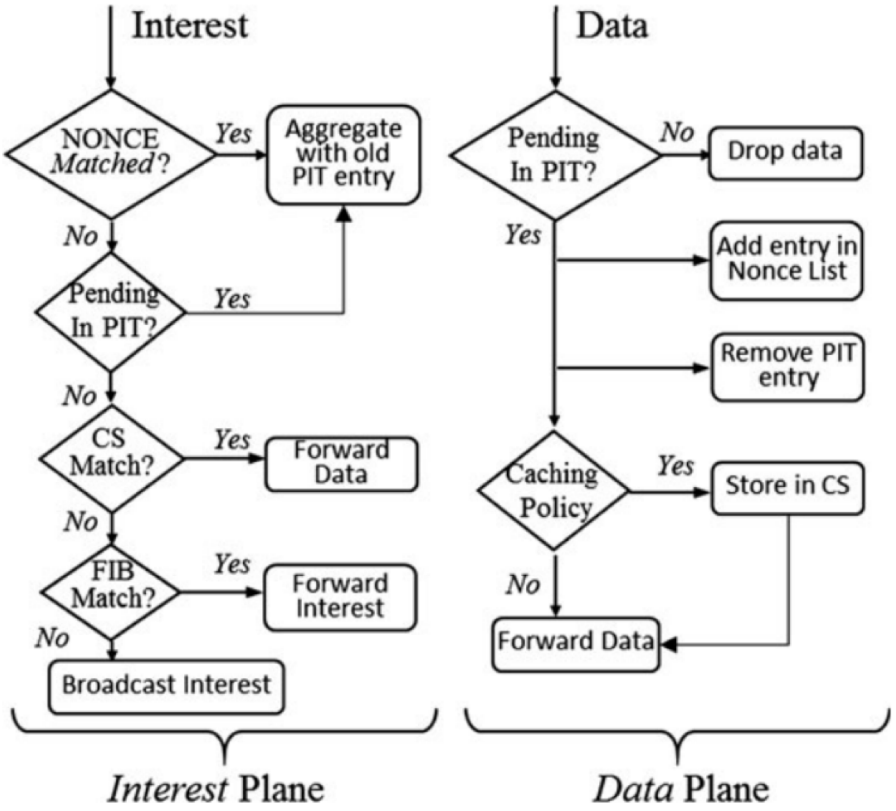

FIGURE 2.4: Flow chart of basic NDN operations.

2.2.1 Naming

Naming in NDN application is the most important part, and NDN names are opaque to the network which allows each application to choose the naming scheme for their needs. There are three naming schemes: flat naming, hierarchical naming and attribute-based naming, where the NDN design considered hierarchically structured names. For example, a video produced by UCLA may have the name /ucla/videos/demo.mpg. This hierarchical structure allows NDN applications to represent the context and relationships of data elements e.g. the name for segment 3 of version 1 of a UCLA demo video might be /ucla/videos/demo.mpg/1/3. Flat names likely useful can be accommodated as a special case in local environments.

In case of dynamically generated data, consumer must deterministically construct the name for desired data chunk without previously seeing the name or data. For finding matching data globally or locally, the globally retrieved data must have globally unique name and local names used for local communication required local routing. Naming enables support for functionality such as content distribution, multicast, mobility, and delay tolerant networking. Namespace management is not part of the NDN architecture. However, enabling application developers or users to design their own namespace for data exchange has several benefits such as: increasing the closeness of mapping between an applications data and its use of network; reducing the need for secondary notation and expanding the range of abstractions available to the developers.

2.3 Data Centric Security

In TCP/IP responsibility for security is left to end points [2]. In contrast, NDN secures the data itself by requiring data producers to cryptographically sign every data packet. The NDN security model assumes that the users know both content name and its publisher's keys, and the publisher signature ensure the integrity and enables determination of data province. Therefore it allows the users to trust in data and the location of the data from where and how the data is obtained is not important anymore.

NDN application can control access to data via encryption and distribute keys as encrypted in NDN data. Requiring signatures on network routing and control messages (like any other NDN data) provides a solid foundation for securing routing protocols against, for example spoofing and tampering. NDN using both multi path and adaptive forwarding strategy model, decreases prefix hijacking because NDN routers can detect anomalies caused by hijacks and retrieve data through alternative paths [3]. Since NDN packets reference content rather than devices, it is trickier to maliciously target a particular device, although mitigation mechanisms will be needed against other NDN- specific attacks, for example, Interest flooding DoS [4].

2.4 Routing and Forwarding

In NDN the routing and forwarding is based on names. In contrast to IP, NDN eliminates three problems caused by addresses in IP architecture such as: address space exhaustion, NAT traversal, and address management. With unbound namespace, there is no address exhaustion. There is no NAT traversal problem since NDN does away with addresses, public or private, and address assignment and management is no longer required in local networks [2].

NDN can use conventional routing algorithm such as link state and distance vector. NDN routers announcing name prefixes instead of IP prefixes and the routing protocol propagates these announcements across the network, informing each routers construction of its own FIB. PIT is responsible for recording each pending interest with its incoming interface(s). In the case of receiving matching data to the router or a time out occurs, PIT removes the pending interest packet. In the FIB Based on information and performance measurement, forwarding strategy modules in each router makes decision on which interest to be forwarded to which interfaces, how many unsatisfied interests to be allowed in PIT, the relative priority of different interests, load balancing interest forwarding among multiple interfaces, and choosing alternative paths to avoid detected failures [2]. If the router decides that the interest cannot be satisfied, there is no forwarding entry happens in FIB, or extreme congestion occurs. The router can send a NACK to its downstream neighbor node that transmitted the interest to forward to other interfaces to explore alternative paths. The PIT enable routers to identify and discard looping packets, it allows them to use multiple paths toward the same data producer.

There are some valuable purposes served by PIT, such as: support multicast delivery, because PIT records the set of interfaces over which the interest for the same name data arrived, traffic load can be controlled. Since each interest retrieves at most one data delivery, by controlling the number of pending interests occurred flow balance, router can control the traffic load. The number of PIT entries shows the router load. By constraining the size, DDoS attack can be limited. PIT entry time out offers cheap attack detection.

2.5 In-Network Storage

NDN routers have a storage. It can be used to cache the popular content to reuse for future incoming Interest. By caching content in NDN routers a significant content delivery delay will be reduced, and content distribution be efficiently achieved. In today's Internet architecture, IP routers are not able to reuse the content after forwarding the content to the requesting party, while NDN routers can. In NDN for static files, NDN achieves optimal data delivery. Also dynamic content can benefit from caching in the case of multicast (e.g, realtime video conferencing) or transmission after a packet loss.

In addition to Content Store, NDN architecture support Repository, a more persistent and larger-volume in-network storage. These services are able to support services similar to Content Delivery Networks (CDN), without engineering them as an application layer to make them work [2].

2.6 Cache Replacement

For efficiency of caching, cache replacement policies play an important role, because the ICN router cache is limited and cannot hold all the content inside the cache. To have some space for new content, cache replacement is required. There are different cache replacement policies. One of the most used and popular policy is Least Recently Used (LRU). This policy removes the least recently used contents from a cache when the cache becomes full [5].

LRU policy is based on the locality of reference seen in request stream which characterizes the future access from the past. There are two types of locality: temporal and spatial. LRU replacement policy considers temporal locality as its main factor; if some contents are recently accessed, the same contents are expected to be accessed again soon in the future. Spatial locality references to some contents which can be nearby reference for other contents in the future. In LRU replacement policy, new incoming requests are inserted at the head of the list, and replacement policy takes place from the end of the list.

Chapter 3

LCD-Based Probabilistic Caching

3.1 Introduction

Today's Internet architecture reveals its inefficiency by the rapid popularization of content-oriented applications like Facebook, YouTube, real-time video streaming, etc. These services deliver a significant amount of data which may cause a long delay for obtaining the content. Cisco Visual Networking Index: Forecast and Methodology 2015 - in the 2020, forecasts that IP video traffic will increase globally to 82 percent of all internet traffic by 2020 [7]. Virtual-reality traffic quadrupled in 2015, from 4.2PB per month in 2014 to 17.9PB per month in 2015. It will increase 61-fold between 2015-2020.

By the tremendous growth of video in the Internet and increasing demands for data access, the question arises on how to efficiently store and distribute these large contents while many end users request the same contents at the same time according to their popularity. The requested contents will repeatedly be transmitted from the servers to the user. It is a waste of bandwidth and increases the user-perceived delay of content delivery. As a future of the Internet architecture, Information-Centric Networking (ICN) has a distinctive feature called *in-network caching* [8][17]. There are many proposals of ICN. Named Data Networking (NDN) is one of the most frequently quoted ICN system at this moment. In NDN, a content is requested using an *Interest packet* the server will return the content with a *Data packet*. By caching the *Data packet* in NDN routers, a subsequent request for the same content needs no longer to traverse the network to the host servers but can easily find the content from a closer NDN router. Therefore,

a significant amount of redundant traffic load can be saved.

Leave Copy Everywhere (LCE) strategy for cache placement and *Least Recently Used* (LRU) for cache replacement are frequently used cache policies [6][18]. In LCE, when a user sends an Interest packet for a content and the content is delivered, a copy of the content will be cached in all intermediate routers. LCE caches many content replicas with every request for the content. Thus, there will be a significant reduction of server load and increase of cache hit if each router has enough amount of cache. Cache storage is limited, however, and LCE causes frequent cache replacement. Leave Copy Down (LCD) and Move Copy Down (MCD) are other cache placement policies. When a user sends an Interest packet, and cache hit occurs, the content will be cached only in the neighbor downstream node. LCD pushes a copy of the content one hop closer to the client after each cache hit [6]. Also in MCD once a cache hit occurs the content is cached only at the neighbor downstream router. MCD deletes the cached content after the hit while LCD does not.

For efficiency of caching, cache replacement policies play an important role. Cache replacement is required to have space for new content because the cache storage in an ICN router is limited. There are different cache replacement policies. One of the most used and popular policy is Least Recently Used (LRU). This policy removes the least recently used contents from a cache when the cache becomes full. A popular content is usually demanded more than a least popular content in the network [5][9]. Least Frequently Used (LFU) strategy uses the frequency of request. LFU removes the least frequently requested contents [9]. First-In First-Out (FIFO) strategy chooses that the oldest content as the one to be removed. It considers the content that entered to cache first will not be used again soon, and removes this content without care about how often or how many times it was accessed before [5][9].

In this Chapter, we propose *LCD-Based Probabilistic Caching* (LBPC), which aims to increase the performance of in-network caching. The goal of LBPC is to cache popular data at the network edge and avoid unpopular data to be cached in the network. To achieve our goal, LBPC caches the content with the highest

probability at the first hop from the server. When a consumer sends an Interest packet for a content for the first time, the first hop router caches the content with a probability of one and cache probability decreased hop by hop. If the same content is requested multiple times, the content is pushed hop by hop towards the user, which can guarantee the popular content are cached in ICN router located closer to the user. The simulation result shows that proposed LBPC can achieve higher cache hit rates compared with *ProbCache*, *LCE* and *LCD*.

3.2 Related Works

There are several studies regarding cache placement strategies for ICN. One of the cache placement policy is ProbCache [11]. This approach approximates the caching capability by considering the cache capacity of ICN routers and the distance from the content server, and caches contents probabilistically on the path. In another work authors proposed HPC [13]. In this policy, each ICN router caches the content probabilistically that is inversely proportional to its distance from data source based on the number of hops and the residence time of cached content.

In both probabilistic strategies, there is a risk that unpopular contents might be cached redundantly in place of popular contents in ICN routers located closer to the user.

In [14] the authors proposed MPC. It counts the number of requests for each content in every ICN router. The router stores the content name and request count into *Popularity Table*. When a request count hits the *Popularity Threshold*, the content name is tagged as popular content, and the router sends a suggestion for neighbor routers to cache the content. As the popularity of a content can decrease with time after the suggestion, the popularity count is reset by *Reset Value* to prevent flooding of the same content for the neighbor node.

In [15] the authors proposed WAVE. At the first request, it divides the content into chunks and distributes content chunks towards the requester by considering the popularity of the content. When the popularity of the content increases, WAVE exponentially enhances the number of chunks to be cached. Upstream routers mark the data packet and suggest its downstream routers cache the chunk. If the downstream router cannot cache, it leaves for another downstream router to cache. Therefore, caching in WAVE does not consider the cache capacity. It prominent in high-level of cache replication near to the producer.

In [16] the authors proposed Centrality-Based Caching. It measures the number of times a router lies on the content delivery path between all the routers in the network and the server with the content. When a router lies with a high number of times on the delivery paths, then it caches. This approach can increase the performance, but the popularity of the content is not considered. In addition, it emphasize particular routers in the network while the cache capacity of others remains unused.

In [17] the authors proposed Congestion-Aware Caching where each ICN router uses a utility function to approximate the value of caching. Whenever a data packet passes through a router, the router calculates the utility value for the content in the data packet. When the utility is higher than any utility values for the contents stored in the cache at the router, the arriving content replaces the one with the lowest utility value.

The authors of [18] proposed CGTIN. In this approach, each user get the flow rate (content request rate) of all nodes on a content delivery path and computes the importance of each node according to the number of requests each node received. When a user sends an Interest packet, the Interest packet records the nodes importance of content delivery path. The importance of the routers are inserted into the header of Interest packet and when producers respond to the Interest packet with a Data packet the importance of routers is copied into the Data packet. After delivery of the content when a cache hit occurs in the intermediate router, which is the most important router, whose request rate is high,

this content will be marked as recently used and not cached at other nodes. Otherwise, if the router is not the most important one, the content will be removed from this router and pushes this content for one level to the downstream routers to cache at a node which just more important node.

3.3 LCD-Based Probabilistic Caching Model

3.3.1 System Model

Each NDN router R_i has a certain limited cache capacity C_i. When a user requests a content, the user sends an Interest packet and the original source of the content or a router caching the requested content responds by replying a Data packet.

The routers on the path of the Data packet are located at a certain distance i from the source in terms of the number of hops. Let us name the router R_i that is located i hops away from the source.

The router R_d with the number of hops d from the source on the path of a Data packet caches the Data packet with probability $P(d)$. When a cache is full, the room for additional content to be cached is created by LRU cache replacement policy. How to compute $P(d)$ is discussed in the next section.

3.3.2 Probability Calculation in LBPC

The total cache capacity of the path from the source of a content to router R_d is

$$\sum_{R_i i \in \Phi_d} C_i$$

where Φ_d is the set of routers on the path from the source to router R_d. The value d can be counted by adding a field to count the number of hops in the header of Data packet. $P(d)$ is calculated in every router R_d that the content pass through. Each NDN router increases the value of the hop count field. $P(d)$ is defined as

$$P(d) = \frac{C_d}{\sum_{R_i i \in \Phi_d} C_i \times d} \tag{3.1}$$

As the value d increases, $P(d)$ decreases.

When the requested content is found in the cache of a router, the hop count field in Data packet is set to zero. The rationale behind this decision is as follows. When there are many requests for the same content, which means high popularity of the content, it is better to push that content closer to the consumer.

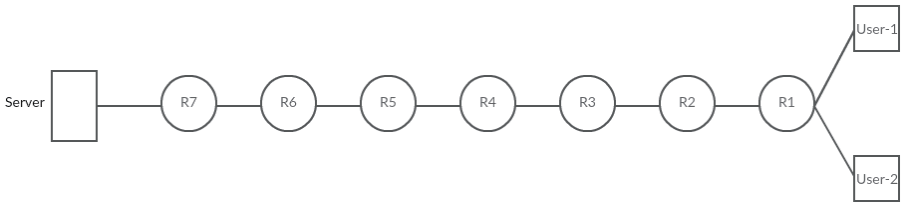

FIGURE 3.1: Linear Topology

For better understanding of the equation 3.1, let us use the example network shown in Fig. 3.1. User 1 sends an Interest packet to retrieve a content held by the server. If there is no content cached in routers on the path, the requests are forwarded to the server. The server responds to the Interest packets with a Data packet. When the Data packet arrives at R_7, R_7 caches the data with 100 % probability as calculated by the equation 3.1, and the cache probability is decreased to (25%, 11%, 6%) in R_6, R_5, and R_4, respectively. When User 2 sends another Interest packet for the same content, R_7 acts as a server of the requested content and R_6 caches the content with 100% probability, and the cache probability decreases in the next hops.

TABLE 3.1: LCD-Based Probabilistic Caching Notation

SYMBOL	MEANING
C_i	Cache capacity of the (R_i).
d	Distance from the content source to R_i.

Suppose there are many requests sent for the same content. The content is pushed to the network edge which is located closer to the users and eventually R_1 caches the content with 100% probability.

3.4 Performance Evaluation

3.4.1 Simulation Set-up

We simulated our algorithm using ndnSIM 2.1 simulator [19]. We compared our proposed algorithm against *ProbCache*, *LCE*, and *LCD*. We used a tree topology shown in Table. 3.2 with 10Mbps link bandwidth and 1ms link delay for our simulation; there are 16 consumers (users), one producer (server), and seven routers. Each consumer sends 4 Interest packet/sec. User requests follow Zipf distribution with α=0.7. The total number of contents in the network is set to 1000 contents. Every NDN router caches the incoming content with the calculated probability. We investigate the cache system performance at different cache memory sizes of 0.5%, 5%, 10%, and 15% of the total number of contents. We simulated our algorithm for 20min.

TABLE 3.2: Simulation Parameters

PARAMETERS	VALUE'S
Total request rate	64 pkt/sec
Total number of contents in the network	1000 packets
Cache memory size of the router	5, 10, 15 (%)
Simulation Time	20min

3.4.2 Simulation Result

For evaluation, we calculated *Cache Hit Ratio* and *Server Load*. Fig. 3.3 shows the simulation result of cache hit ratio. Leaf level routers in Fig. 3.2 have cache capacity of 15% of the total contents, rtr-1, and rtr-2 have cache capacity of 10%, and rtr-0 has cache capacity of 5%. After 20min of simulation, the result shows LBPC has better performance with 42% cache hit ratio at leaf routers, compared with cache hit ratios of ProbCache, LCE and LCD which are 36.08%, 35.17%, and 34.56%, respectively.

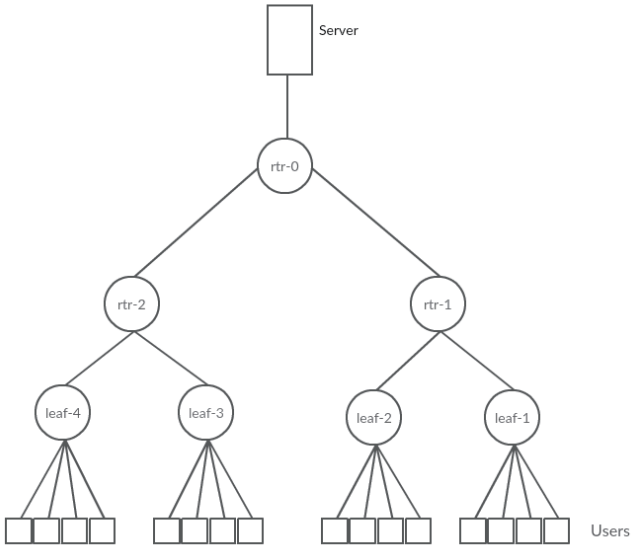

FIGURE 3.2: Tree Topology.

Fig. 3.4 shows the result of cache hit ratio, where leaf level nodes have cache capacity of 5%, rtr-1, and rtr-2 have cache capacity of 10%, and rtr-0 has cache capacity of 15%. The result shows LBPC has higher cache hit ratio 25% at leaf routers, compare with ProbCache, LCE, and LCD which are 17.41%, 16.13%, and 11.38% respectively. In intermediate nodes rtr1 and rtr2, LCD has better performance compare with LBPC, ProbCache, LCE and LCD policies.

Fig. 3.5 shows the result of cache hit ratio when leaf nodes, the intermediate nodes and rtr-0 have cache memory size of 5%, 15%, and 10% of the total number of contents respectively. The simulation result shows LBPC has average cache hit ratio of 25% in leaf routers, compare with the cache hit ratios ProbCache, LCE, and LCD which are 17%, 16%, and 11%. respectively.

The result in Fig. 3.3 shows that with larger cache memory size at leaf routers, more requests are satisfied at leaf routers. While, smaller cache memory at leaf routers has less cache hit performance as shown in Fig. 3.4 and Fig. 3.5. Moreover, LBPC pushes the popular content faster to the edge routers, most popular contents are cached at the edge which accounts for most of the cache hits; less popular contents might be cached at intermediate routers far away from consumers.

In the case of small cache memory size allocation at intermediate routers, leaf routers, the intermediate routers, and rtr-0 have cache memory size of 15%, 0.5%, and 15% of the total number of contents, respectively. Only in this cache memory size allocation shown in Fig. 3.6, LBPC had lower cache hit with 32% compared with cache hit ratios of *ProbCache*, *LCE*, and *LCD* which are 36%, 35% and 35%, respectively at leaf routers. LBPC is based on LCD. Because of small cache memory size assigned in intermediate routers, most of the popular content cannot be cached at intermediate nodes to push these contents to the routers which are closer to the consumer. Therefore, cache hit ratio in LBPC is decreased. However, LBPC has better performance of total cache hit ratio in the network compare with ProbCache, LCE and LCD.

In Fig. 3.7 the graph shows server load with different cache memory sizes. The leaf routers, the intermediate routers, and rtr-0 have cache memory size of 15%, 10%, and 5%, respectively. LBPC causes lower server load than ProbCache, LCE, and LCD.

For other cache memory size allocation, LBPC also lowers server load as shown in Fig. 3.8 and Fig. 3.9 except for the case where the intermediate routers are assigned the cache capacity of 0.5% of the total contents as shown in Fig. 3.10. Our simulation result in Fig. 3.10 shows ProbCache, LCE and LCD has less server load against LBPC.

Fig. 3.11 shows the result of average hop count with cache memory size of the leaf routers, the intermediate routers, and rtr-0 15%, 10%, and 5%, respectively. The simulation result shows LBPC has shorter average hop count of 2.61compare with ProbCache, LCE and LCD where the average hop counts are 2.81, 2.83, and 2.86.

For other cache allocation, LBPC results shorter average hop counts than ProbCache, LCE, and LCD as shown in Fig. 3.12 and 3.13 except for the case where the intermediate routers (rtr-1, rtr-2) have cache capacity 0.5%, the leaf routers have cache memory size of 15%, and the rtr-0 node has cache memory size of 15%. Our simulation result shown in Fig. 3.14 shows ProbCache, LCE and LCD has shorter average hop counts of 2.83 compare with LBPC, LCD, ProbCache and

HPC where the average hop counts are 2.89, 2.94, 4, 4, respectively.

Because LBPC caches more popular contents at the network edge as shown in Fig. 3.3, Fig. 3.4, and Fig. 3.6, more requests are satisfied at leaf routers and sub sequent requests for the same content needs no longer to traverse the network to the host server. Therefore, LBPC has better performance with a significant reduction of server load and shorter average hop count. In the case of small cache memory size allocation at intermediate routers shown in Fig. 3.10, the cache hit ratio is decreased in LBPC. it affects the performance of server load and hop count as well. ProbCache, LCE and LCD have more reduction of server load and shorter average hop count compares with LBPC.

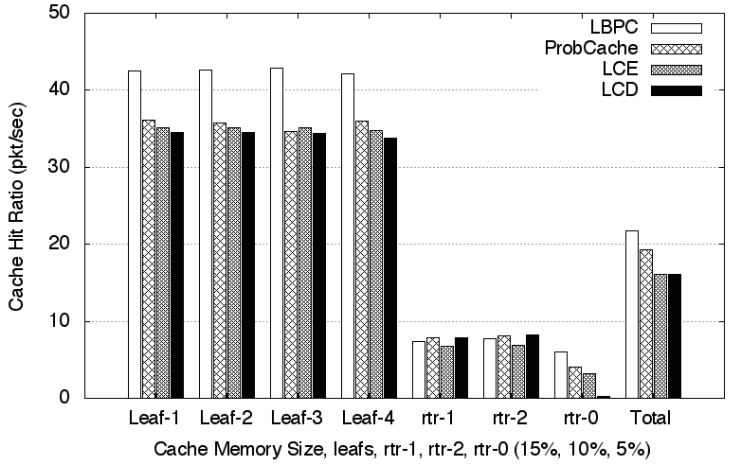

FIGURE 3.3: Cache hit ratio with different Cache Memory size of routers. leafs, rtr-1,2, rtr-0 (15%, 10%, 5%)

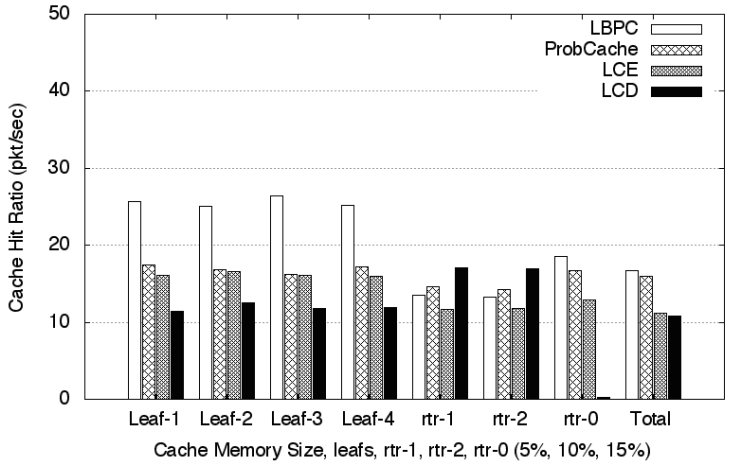

FIGURE 3.4: Cache hit ratio with different Cache Memory size of routers. leafs, rtr-1,2, rtr-0 (5%, 10%, 15%)

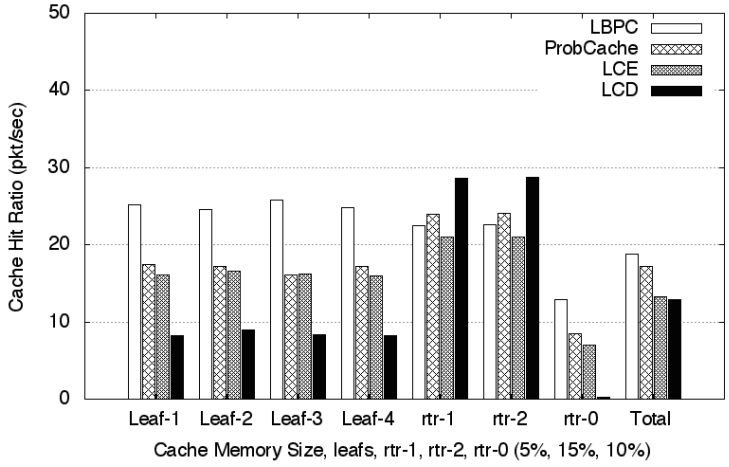

FIGURE 3.5: Cache hit ratio with different Cache Memory size of routers. leafs, rtr-1,2, rtr-0 (5%, 15%, 10%)

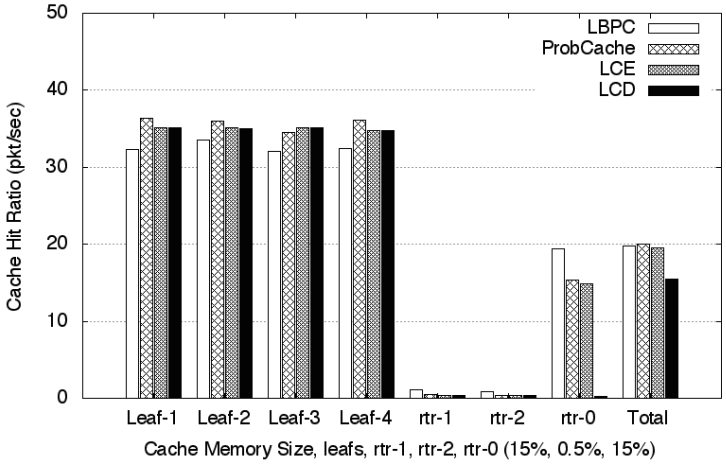

FIGURE 3.6: Cache hit ratio with different Cache Memory size of routers. leafs, rtr-1,2, rtr-0 (15%, 0.5%, 15%)

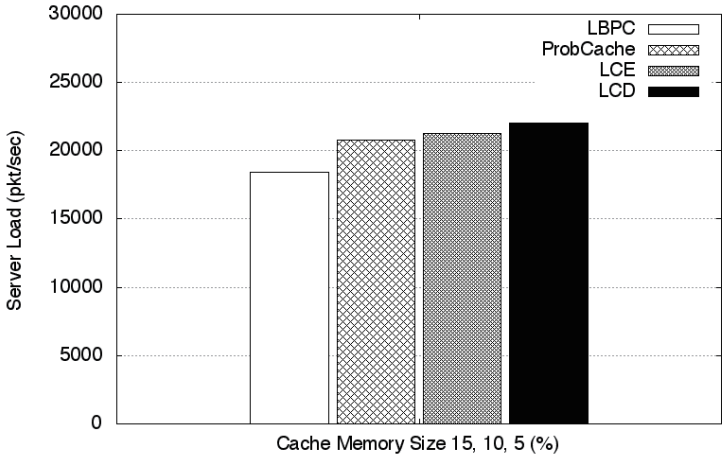

FIGURE 3.7: Server load different with Cache Memory size of routers. leafs, rtr-1,2, rtr-0 (15%, 10%, 5%)

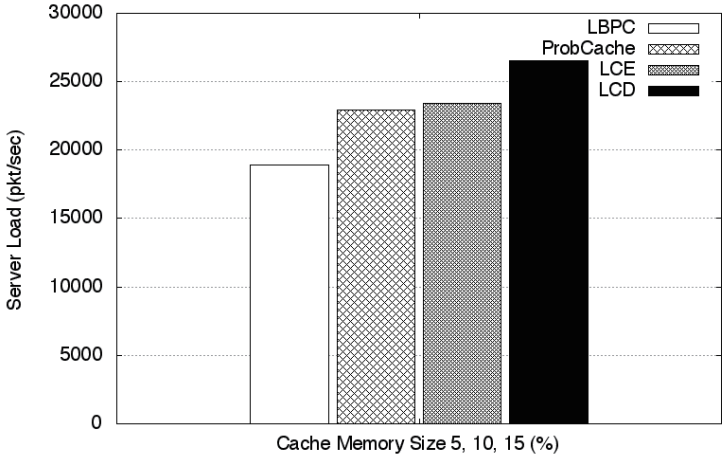

FIGURE 3.8: Server load with different Cache Memory size of routers. leafs, rtr-1,2, rtr-0 (5%, 10%, 15%)

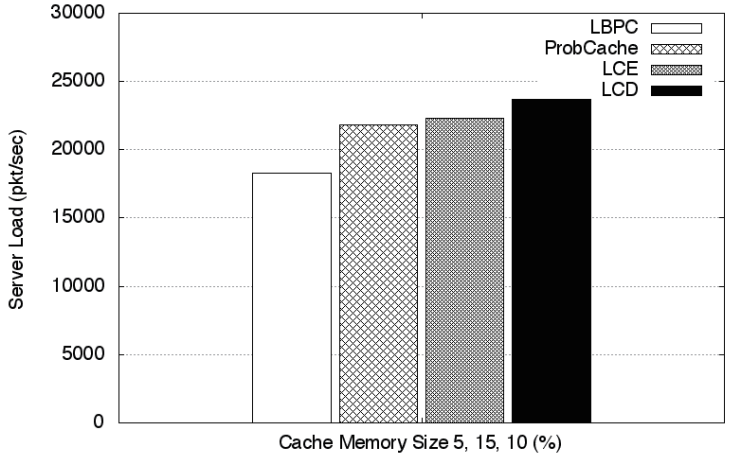

FIGURE 3.9: Server load with different Cache Memory size of routers. leafs, rtr-1,2, rtr-0 (5%, 15%, 10%)

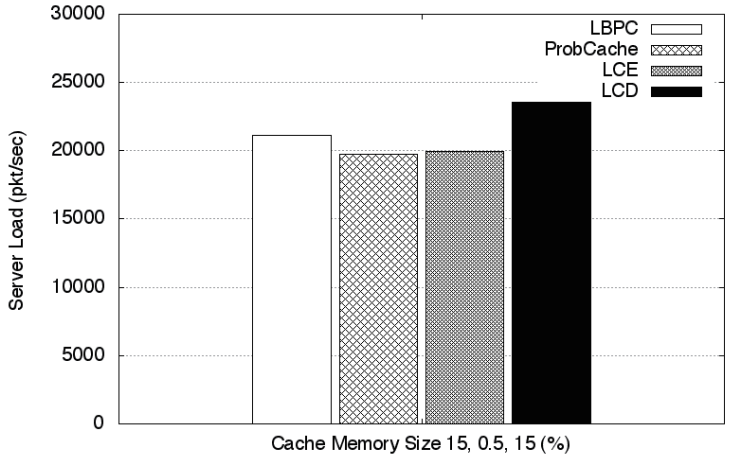

FIGURE 3.10: Server load with different Cache Memory size of routers. leafs, rtr-1,2, rtr-0 (15%, 0.5%, 15%)

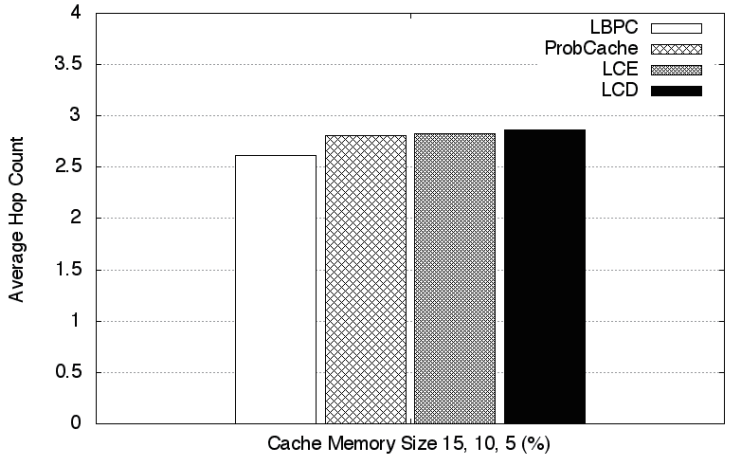

FIGURE 3.11: Hop Count with different Cache Memory size of routers. leafs, rtr-1,2, rtr-0 (15%, 10%, 5%)

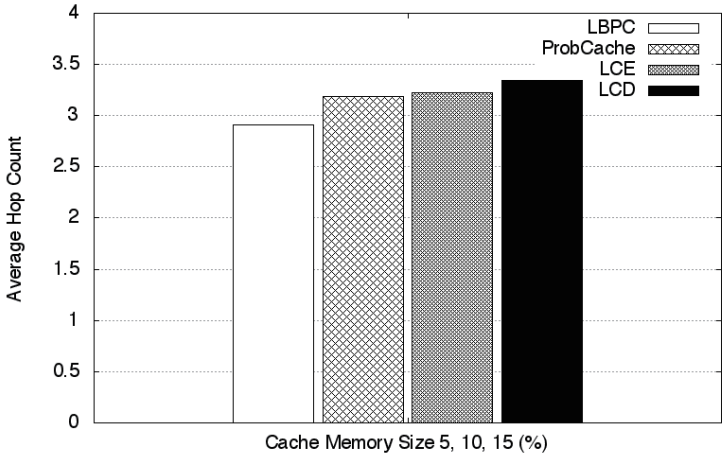

FIGURE 3.12: Hop count with different Cache Memory size of routers. leafs, rtr-1,2, rtr-0 (5%, 10%, 15%)

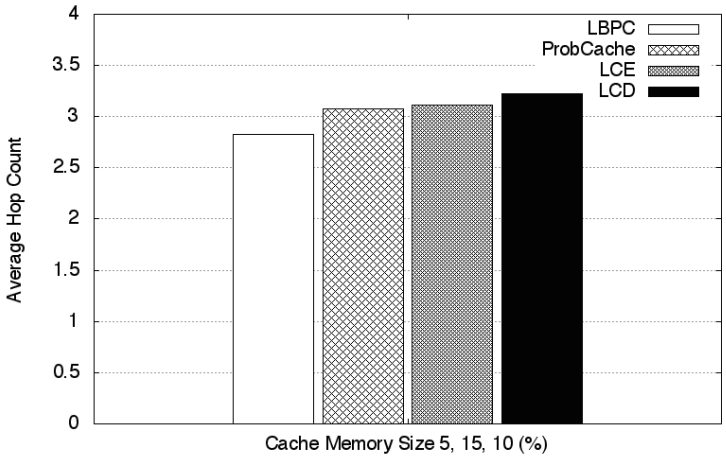

FIGURE 3.13: Hop count with different Cache Memory size of routers. leafs, rtr-1,2, rtr-0 (5%, 15%, 10%)

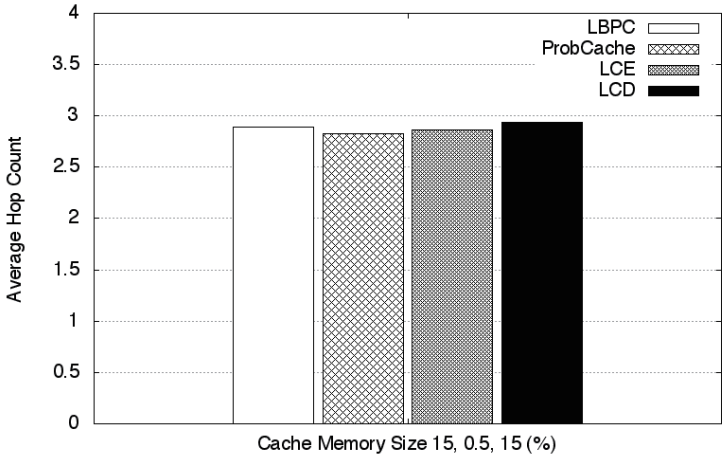

FIGURE 3.14: Hop count with different Cache Memory size of routers. leafs, rtr-1,2, rtr-0 (15%, 0.5%, 15%)

Chapter 4

LCD-Based Probabilistic Edge Caching

4.1 Introduction

In this chapter we try to solve the cache problems that occurred in Chapter 3. In the case of small cache memory size allocation at intermediate routers, LBPC has less cache hit ratio compared with existing proposed algorithms such as *ProbCache*, *LCE*, and *LCD*.

For improving the LBPC, we propose a new cache scheme, named as *LCD-Based Probabilistic Edge Caching* (LBPEC). LBPEC has higher cache hit ratio to efficiently reduce server load and hop count. LBPEC caches the content with the highest probability at the two end routers which are located closer to the server and user. when a requested content send for the first time, the first hop router closer to the server and user caches the content with a probability of one.

The simulation result shows that proposed LBPEC can achieve higher cache hit rates compared with LBPC, ProbCache, LCE, and LCD.

4.2 LCD-Based Probabilistic Edge Caching Model

4.2.1 System Model

Each NDN router i has a certain limited cache capacity C_i. When a user requests a content, the user sends an Interest packet and the original source of the content

or a router caching the requested content responds by replying a Data packet.

Where c is the number of hops of the path, the value c can be counted by adding a field to count the number of hops in the header of Interest packet. d is the number of hop from producer. The notation is summarized in Table. 4.1. When the server responds to the Interest packet with a Data packet, c value remain as fixed value, while d value is increased hop by hop.

The router i caches the Data packet with probability $P(d)$. How to compute $P(d)$ is discussed in the next section.

4.2.2 Probability Calculation in LBPEC

Each router along the path caches the content with probability $P(d)$, depending on the value c and d. The calculation of $P(d)$ is defined as

$$P(d) = \frac{C_d \times c}{\sum_{i=1}^{c-(d-1)} C_i \times d} \tag{4.1}$$

For better understanding of the equation, lets assume User-1 and User-2 in Fig. 4.1 sending an Interest packet to retrieve the content, if there is no cached content in any router, the number of hops in the path is $c = 7$. When server respond the Interest packet with Data packet, the value of $d = 1$ initially, and hop by hop d value is increasing. The probability density function graph of the equation 4.1 is shown in Fig. 4.2.

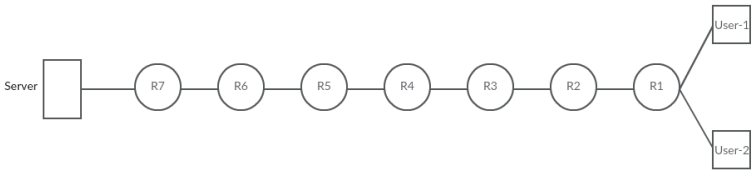

FIGURE 4.1: Linear Topology

When the requested content is found in the cache of a router, the router acts as a server and d value is set to one. c value is replaced by that of the Interest packet. In this way, if there are many requests for the same content, which means high popularity of the content, it push that content closer to the consumer.

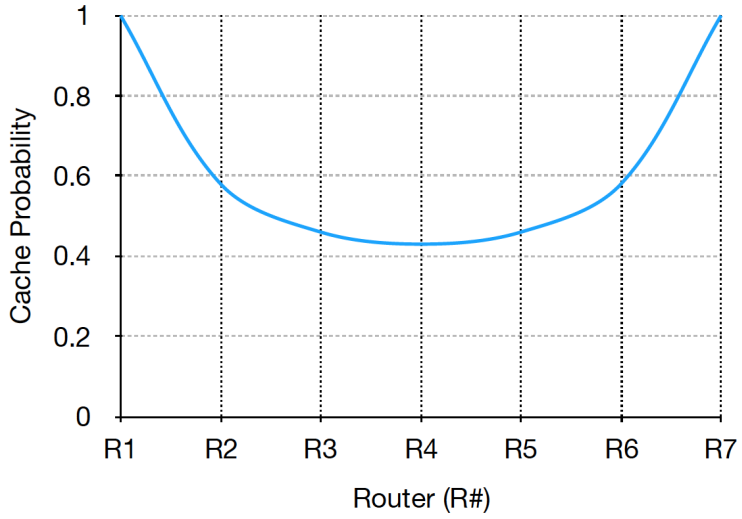

FIGURE 4.2: Probability density function graph

TABLE 4.1: LCD-Based Probabilistic Edge Caching Notation

SYMBOL	MEANING
C_i	Cache capacity of the (R_i).
c	Hop distance from the consumer.
d	Hop distance from the server.

Suppose there are many requests sent for the same content. The content is pushed to the network edge which is located closer to the users.

4.3 Performance Evaluation

4.3.1 Simulation Set-up

We simulated our algorithm using ndnSIM 2.1 simulator [19]. We compared our proposed algorithm against the previous work *LBPC* [15], *ProbCache*, *LCE*, and *LCD*. We used a tree topology shown in Fig. 4.3 with 10Mbps link bandwidth and 1ms link delay for our simulation; there are 16 consumers (users), one producer (server), and seven routers. Each consumer sends 4 Interest packet/sec. User requests follow Zipf distribution with α=0.7. The total number of contents in the

network is set to 1000 contents. Every NDN router caches the incoming content with the calculated probability. We investigate the cache system performance at different cache memory sizes of 0.5% and 15% of the total number of contents. We simulated our algorithm for 20min.

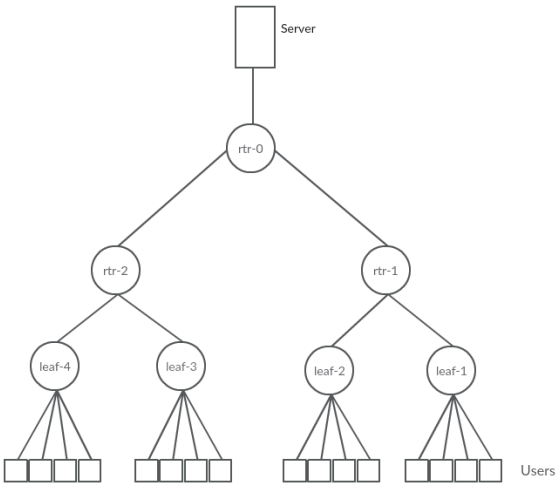

FIGURE 4.3: Tree Topology

4.3.2 Simulation Result

For evaluation, we calculated *Cache Hit Ratio*, *Server Load* and *Hop Count*. Fig. 4.4 shows the simulation result of cache hit ratio. Leaf level routers in Fig. 4.3 have cache capacity of 15% of the total contents, rtr-1, and rtr-2 have cache capacity of 5%, and rtr-0 has cache capacity of 15%. After 20min of simulation, the result shows LBPEC has better performance with 39% cache hit ratio in leaf routers compared with cache hit ratios of LBPC, ProbCache, LCE, and LCD which are 32%, 36%, 35%, and 35%, respectively.

In Fig. 4.5 the graph shows server load with different cache memory sizes. The leaf routers, the intermediate routers, and rtr-0 have cache memory size of 15%, 0.5%, and 15%, respectively. LBPEC has better performance compare with LBPC, ProbCache, LCE, and LCD.

Fig. 4.6 shows the result of average hop count with cache memory size of the leaf nodes, the intermediate nodes, and rtr-0 15%, 0.5%, and 15%. respectively.

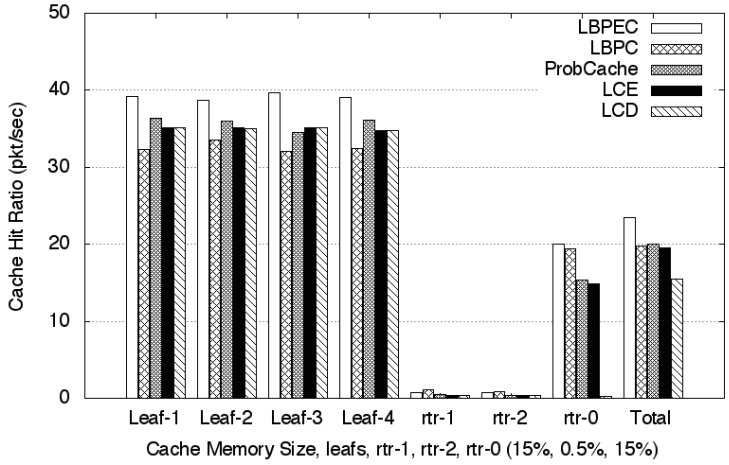

FIGURE 4.4: Cache hit ratio with different Cache Memory size of routers. leafs, rtr-1,2, rtr-0 (15%, 0.5%, 15%)

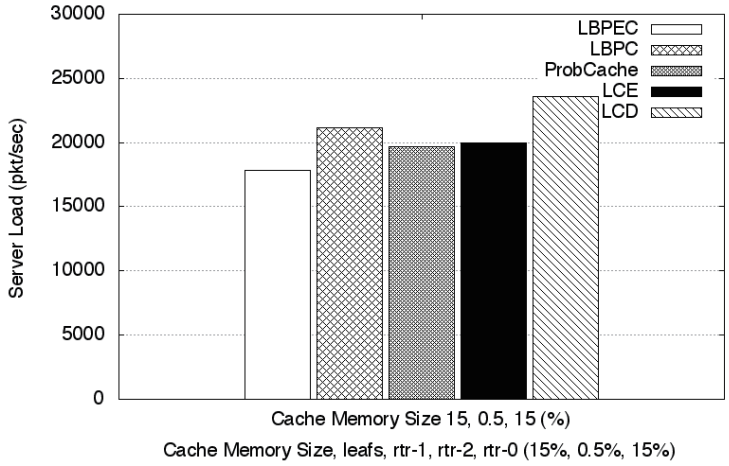

FIGURE 4.5: Server load with different Cache Memory size of routers. leafs, rtr-1,2, rtr-0 (15%, 0.5%, 15%)

The simulation result shows LBPEC has shorter average hop count of 2.69 compare with LBPC, ProbCache, LCE and LCD where the average hop counts are 2.89, 2.83, 2.86, and 2.94.

In the case of small cache memory size allocation at intermediate routers LBPC [20] has lower cache hit ratio. While, LBPEC caches the content with higher probability at the edge network closer to the consumer and server, it pushes the popular content faster to the edge routers, most popular contents are cached at the edge which accounts for most of the cache hits; less popular contents might be cached at intermediate routers far away from consumers. It is because cache

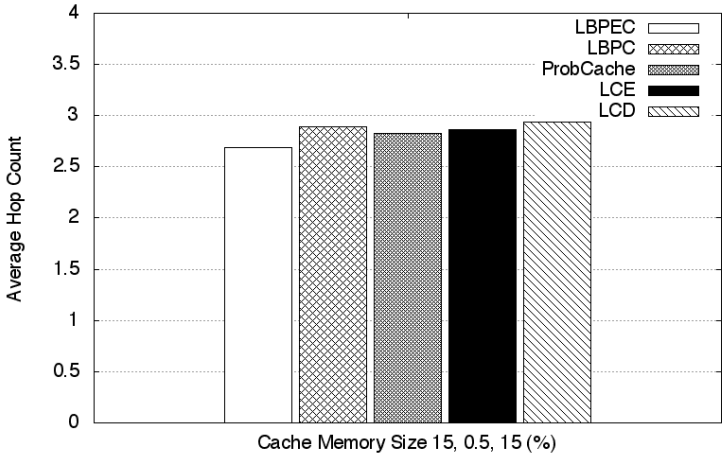

FIGURE 4.6: Hop count with different Cache Memory size of routers. leafs, rtr-1,2, rtr-0 (15%, 0.5%, 15%)

memory size of the router has relation with probability allocation, because the cache memory size in leaf level routers are higher compare with intermediate routers and LBPEC has higher probability allocation at the edge nodes, therefore, the performance of LBPEC is better compare with other proposed algorithms.

Chapter 5

Conclusion

In NDN, there are different research topics. In this thesis we focus on *in-network caching*. Where, ICN routers are able to cache the content to satisfy the requested content from nearest router. There are many algorithms proposed for cache placement policy such as *ProbCache, LCE, and LCD*, We developed a new algorithm called LCD-Based Probabilistic Caching in Chapter 3, which caches contents with the highest probability when the router is located near to the server. After receiving multiple requests for the same content, the content is pushed near to the user hop by hop. The simulation result shows good performance of LCD-Based Probabilistic Caching compare with ProbCache, LCE, and LCD.

In Chapter 4 we developed LCD-Based Probabilistic Edge Caching (LBPEC) to solve the cache problems that occurred in Chapter 3. LBPEC cache the content with higher probability at two edge routers which are located closer to the user and server. In the case of small cache memory size allocation at intermediate nodes LBPC has lower cache hit ratio compare with existing algorithms such as *ProbCache, LCE, and LCD*. While LBPEC cache hit is increased compare with others. It is because cache memory size of the router has relation with probability allocation, because the cache memory size in leaf level nodes are higher compare with intermediate nodes, if we have higher probability allocation at leaf level nodes, the cache hit performance is increased.

Reference

[1] B. Ahlgren, C. Dannewitz, C. Imbrenda, and D. Kutscher, "A Survey of Information-Centric Networking," vol. 16, no. July, pp. 26-36, 2012.

[2] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named Data Networking," vol. 2, no. 1, pp. 66-73, 2014.

[3] C. Yi, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, "Adaptive forwarding in named data networking," ACM SIGCOMM Comput. Commun. Rev., vol. 42, no. 3, p. 62, 2012.

[4] N. Chhetry and H. K. Kalita, "Interest Flooding Attack in Named Data Networking : A Survey," vol. 4, no. 1, 2016.

[5] S. Podlipnig and L. Boszormenyi, "A survey of Web cache replacement strategies," ACM Comput. Surv., vol. 35, no. 4, pp. 374-398, 2003.

[6] N. Laoutaris, S. Syntila, and I. Stavrakakis, "Meta algorithms for hierarchical Web caches," Performance, Comput. Commun. 2004 IEEE Int. Conf., pp. 445-452, 2004.

[7] Cisco, "Cisco Visual Networking Index: Forecast and Methodology, 2015-2020," Forecast Methodol., p. 22, 2015.

[8] V. Jacobson, D. K. Smetters, J. D. Thornton, M. Plass, N. Briggs, and R. Braynard, "Networking named content" Commun. ACM, vol. 55, no. 1, p. 117, 2012.

[9] S. Arif, S. Hassan, and I. Abdullahi, "Cache Replacement Positions in Information-Centric Network," Int. Conf. Internet Appl. Protoc. Serv. (NETAPPS 2016), no. December, pp. 54-58, 2016.

[10] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named Data Networking," vol. 2, no. 1, pp. 66-73, 2014.

[11] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," Proc. Second Ed. ICN Work. Information-centric Netw. - ICN '12, p. 55, 2012.

[12] G. Zhang, Y. Li, and T. Lin, "Caching in information centric networking: A survey," Comput. Networks, vol. 57, no. 16, pp. 3128?3141, 2013.

[13] Y. Wang, M. Xu, and Z. Feng, "Hop-based Probabilistic Caching for Information -Centric Networks," pp. 2102-2107, 2013.

[14] T. Silverston and O. Festor, "MPC": Popularity-based Caching Strategy for Content Centric Networks," pp. 3619-3623, 2013.

[15] K. Cho, M. Lee, K. Park, T. T. Kwon, and Y. Choi, "WAVE": Popularity-based and Collaborative In-network Caching for Content-Oriented Networks," pp. 316-321, 2012.

[16] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache ' Less for More ' in Information-centric Networks."

[17] I. Networks, M. Bay, J. Kurose, and V. Firoiu, "Congestion-Aware Caching and Search in Categories and Subject Descriptors," pp. 37-46.

[18] Y. H. Z. N. S. Zhu, "A Cache Strategy in Content-centric Networks Based on Node's Importance." 2014.

[19] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM 2 . 0: A new version of the NDN simulator for NS-3," pp. 1-8, 2015.

[20] Siddique Abdul Milad and Hidenori Nakazato, LCD-Based Probabilistic Caching for Information Centric Networking, Technical Report of IEICE, CS2017, July 2017.