In-network Caching Strategies for Streaming Content Delivery over Information-Centric Networking

ICN におけるストリーミングコンテンツ配信のインネットワークキャッシング方式

2018 年 2 月

Haipeng LI
李海鵬

In-network Caching Strategies for Streaming Content Delivery over Information-Centric Networking

ICN におけるストリーミングコンテンツ配信のインネットワークキャッシング方式

2018 年 2 月

早稲田大学大学院国際情報通信研究科
国際情報通信学専攻　　分散コンピューティングシステム研究Ⅱ

Haipeng LI
李海鵬

Acknowledgements

First and foremost, I wish to express my sincere gratitude to my supervisor, Prof. NAKAZATO, for the continuous support throughout my Ph.D. study and related researches. His guidance has been of invaluable help in the research and writing of this dissertation, and I appreciate all his contributions of time, ideas, and kindly support to me.

I would also like to express my special gratitude to Prof. TUSDA and other Professors of GIST, Waseda University. I have benefited a lot of valuable guidance from them while we discussed in the research seminar every week or in private talks.

My sincere gratitude also goes to my fellow labmates, Huan WANG, Hidehiro KANEMITSU, Rully ACHMAD and Hongguang QI for the stimulating discussions, for the encouragement, and for the time they spent working with me. I would also like to thank my friends Di ZHANG and Keping YU, who studied in GITS Waseda University. They help me a lot on my researches and publications.

Last but not the least, I would like to thank my family. My father, Xianmin LI, my mother, Cuifang XIAO, who always support me in any situation, encourage me when I fall, and guide me with their wisdom. My Wife, Wenjing YAN, who just got her Ph. D. in Auburn University this year. We have supported each other during our doctoral study.

Contents

List of Figures

List of Tables

List of Abbreviations

AS	Autonomous System
CCN	Content-Centric Networking
CS	Content Store
CB	Candidate Buffer
CVNI	Cisco Visual Networking Index
DONA	Data-Oriented Network Architecture
DFLP	Dynamic File Level Popularity
FIFO	First-In and First-Out
FLCD	Fast Leave Copy Down
FIB	Forwarding Information Base
ICN	Information Centric Networking
IRA	Independent Reference Assumption
LCD	Leave Copy Down
LCE	Leave Copy Everywhere
LRU	Least Recently Used
LFU	Least Frequently Used
NetInf	Network of Information
NDN	Named Data Networking
NRS	Name Resolution System
OPT	Optimal replacement algorithm
PURSUIT	Publish-Subscribe Internet Technologies
PIT	Pending Interest Table
PPCSA	Popularity Proportional Cache Size Allocation

PN	Pre-Name
RXI	Request Expectation Index
RH	Resolution Handler
SAIL	Scalable and Adaptive Internet Solutions
SN	Sequence Number
TTH	Time to Hold
TLP-TTH	Two-Level Popularity-oriented Time-To-Hold
TM	Topology Manager
UWS	Update Window Size

Chapter 1

Introduction

In this chapter, I initially introduce the context of this study. To begin with, the research background is introduced. Furthermore, the current research projects of ICN and the studies of in-network caching are illustrated. In addition, according to the problems that the future Internet may face while delivering streaming content, the research motivation of this study is elaborated, as well as the research aim. Besides, the organization of this dissertation is represented at the end of this chapter.

1.1 Research Background

The Internet has already become one of the most indispensable components of the modern information society. By supporting the unprecedented worldwide communication and the global information sharing, the Internet has greatly changed the pattern of people's lives and has turned into the most important infrastructure of current world in recent decades.

There is no doubt that the Internet evolves according to the improvement of our society. With the great increment of the network scale, the requirement of the Internet has gradually developed towards intelligent, multimedia, high-mobility and energy efficient. Nowadays, most of the content over the Internet is information-intensive, such as video, audio, streaming file, etc. Thus the great development of the current Internet leads to the significant increment of network data volume. According to the latest white paper of Cisco Visual Networking Index (CVNI), the global Internet traffic rises exponentially. The white paper point out that By 2021, the global IP traffic will grow up to 3.3 ZB per year, while 1.2 ZB per year in 2016, and the IP video traffic will account for 82% of all the Internet traffic by 2021 [1, 2]. Indeed, the main use of the Internet has gradually shifted from connection-driven communication, such as host-to-host conversations, to the current widely adopted information-driven communication such as huge amount of data broadcasting and retrieval, as shown in Figure 1.1. People no longer care much about where the data actually be stored, but focus on the information they require. The current IP based network, which was initially designed for the conversation-based communication in the 1960s, is therefore already incommensurate with the current network requirement.

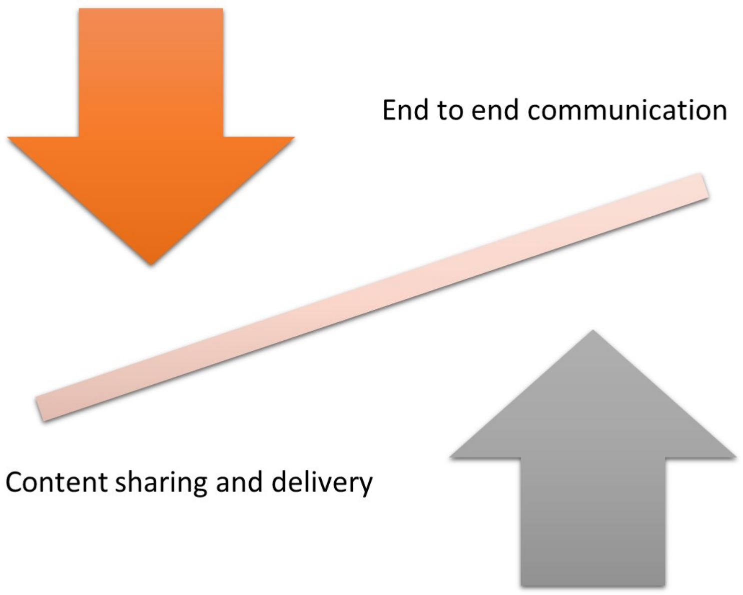


FIGURE 1.1: The change of network requirement.

To respond to this challenge, researchers are dedicated to designing the information-oriented network architectures, i.e. ICN [3, 4]. The basic idea of ICN architecture has been initially introduced in IETF by TRIAD and Baccala in 2002 [7–9]. Unlike IP based network, ICN is an attractive future network architecture that enables transferred information to be named uniquely at the network layer in order that the named content can be responded directly without any location information, namely, the IP address. Apparently, compared with the IP-based network, ICN is more direct and effective for content delivery. In addition, name-based routing also enables the deployment of in-network caching, multi-path routing and content multi-casting. Accordingly, ICN is capable of facilitating data transmission efficiently.

1.2 Overview of ICN Architecture

In recent years, many research projects have been involved in studying ICN-based internet architectures [25–32, 61–70]. Data-Oriented Network Architecture (DONA) from UC Berkeley is one of the first completed ICN architectures, which were proposed in 2007. Subsequently, several research projects have been funded to further investigate the ICN-based network architectures, such as Network of Information (NetInf), Publish-Subscribe Internet Technologies (PURSUIT), Content-Centric Networking (CCN)/Named Data Networking (NDN) [9–14]. These projects are introduced in detail respectively as follows.

1.2.1 DONA

DONA has been proposed by UC Berkeley in 2007, which is the initially proposed and completed ICN architectures. In DONA a flat and self-certifying naming scheme with a hierarchical resolution infrastructure named Resolution Handlers (RHs) is adopted. Accordingly, DONA initially enables the name-based routing and content-level data communication over the Internet.

Every data content in DONA has a unique name called *Principle*. The *Principle* in DONA is structured as $P : L$, where P is the cryptographic hash of the public key, while L uniquely identifies the information with respect to the *Principle*. The *Principle* is globally unique, persistent, and not bounded by any organizational boundaries.

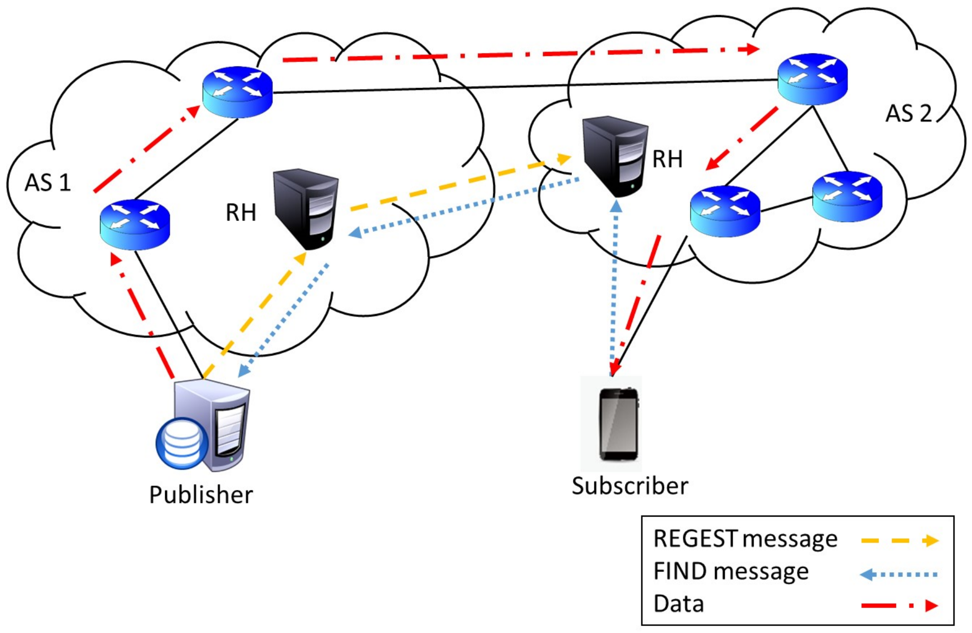


FIGURE 1.2: DONA architecture.

Name resolution in DONA is provided by RHs, at each Autonomous System (AS). And the interconnected RHs which are allocated in different ASs enable the name resolution and between ASs. In DONA, two types of messages: $REGISTER(P : L)$ message, $FIND(P : L)$ message, are adopted by the RHs for the name-based communication. $FIND(P : L)$ message is used for locating the target data with the name $P : L$, while $REGISTER(P : L)$ message is for making a data available and setting up necessary states in the RHs to route subsequent $FIND(P : L)$ messages effectively, as shown in Figure 1.2. In addition, according to the content name routing between the RHs can be performed directly. Once a $FIND(P : L)$ message is resolved, content can be delivered to the client by sending it over the reverse of the appended path. Alternatively, the target content can be responded by IP based routing.

1.2.2 NetInf

NetInf is a part of the EU FP7 projects 4WARD and Scalable and Adaptive Internet Solutions (SAIL) [15–17]. The 4WARD project is more focused on naming and content searching, while the SAIL project focuses on network transport issues.

NetInf uses flat name structure as $ni : //A/L$ in which the names composed with A and L. A is the hash of publisher's public key and L is a label chosen by the publisher. In NetInf, content routing and content name resolution can be either decoupled or coupled.

For the scenario that the name resolution and data routing are decoupled, Name Resolution System (NRS) is adopted. NRS is used for map the content name to the locators to reach the corresponding content, and a multilevel DHT-based name resolution service called MDHT [18] is used to provides name-based routing. At each AS, the local NRS is in charge of the name resolution for L, and the global NRS for the A. When a publisher wants to publish a content, a PUBLISH message with a L is sent to the NRS in the same AS to locator mapping. All the Ls for the same authority are aggregated by the local NRS, and this information is sent to the global NRS through a PUBLISH message. and a PUBLISH message is sent to the global NRS. And the global NRS update the latest mapping according to the received PUBLISH message. When a subscriber wants to request a content, the subscriber can sent a GET message to the local NRS, and a locator for target content will be return by the local NRS after consults with the global NRS. Finally, by using the returned locator, a GET message is sent to the publisher by the subscriber, and the publisher responds with a DATA message with the requested content. The processing is shown in Figure 1.3.

In the scenario that the name resolution and data routing are coupled, a routing protocol is adopted to manage the content retrieval. When a subscriber wants a content, subscriber have to send a GET message to the local CR, and this GET message is forwarded hop-by-hop in order to approach to the publisher or a cache. A DATA message is responded when it's found through the path taken by the previous GET message.

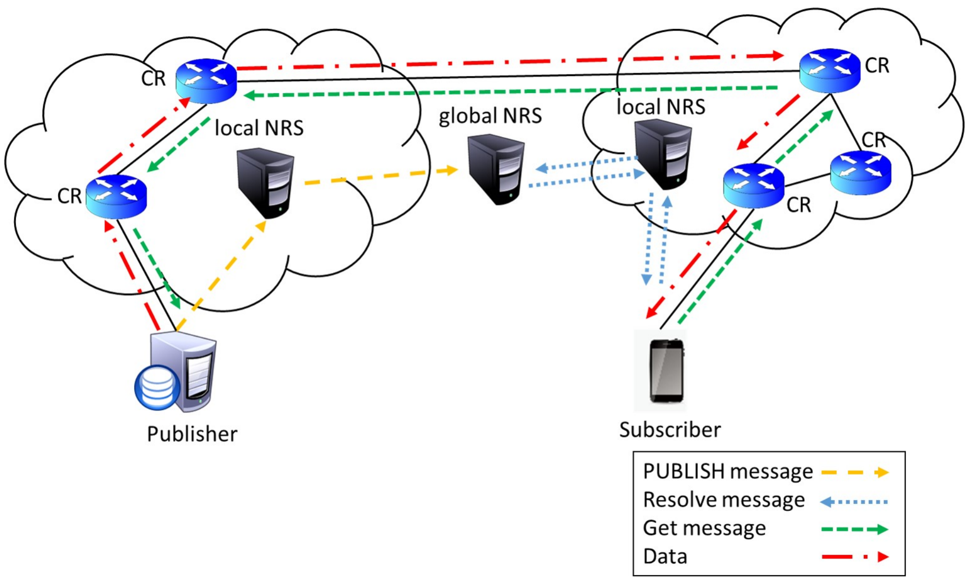


FIGURE 1.3: NetInf architecture.

1.2.3 PSIRP/PURSUIT

PURSUIT is a follow-up research of PSIRP which is a former EU FP7 project. Unlike IP based Internet routing architecture, PSIRP proposed a publish-subscribe paradigm based routing architecture.

The content name used in PURSUIT, i.e. resource identifiers (RIds), is composed of the scope ID and the rendezvous ID. The scope ID if for identifying a set of related contents, and the rendezvous ID is for distinguishing a particular segment of content.

In PURSUIT, the Rendezvous Nodes (RNs) are adopted for the name resolution. When a publisher wants to publish a content, a PUBLISH message is sent to the local RN. And if a subscriber wants the same content, he should send a SUBSCRIBE message to the local RN, and this massage will be forwarded by the DHT to the target RN. After receives the SUBSCRIBE message, the RN sends a massage to the node,namely Topology Manager (TM), in order to get a route from the publisher to the subscriber. Subsequently, the TM sends a START PUBLISH message to the publisher which including this route information, and the target content is retrieved hop-by-hop through Forwarding Nodes (FNs). The processing is in Figure 1.4.

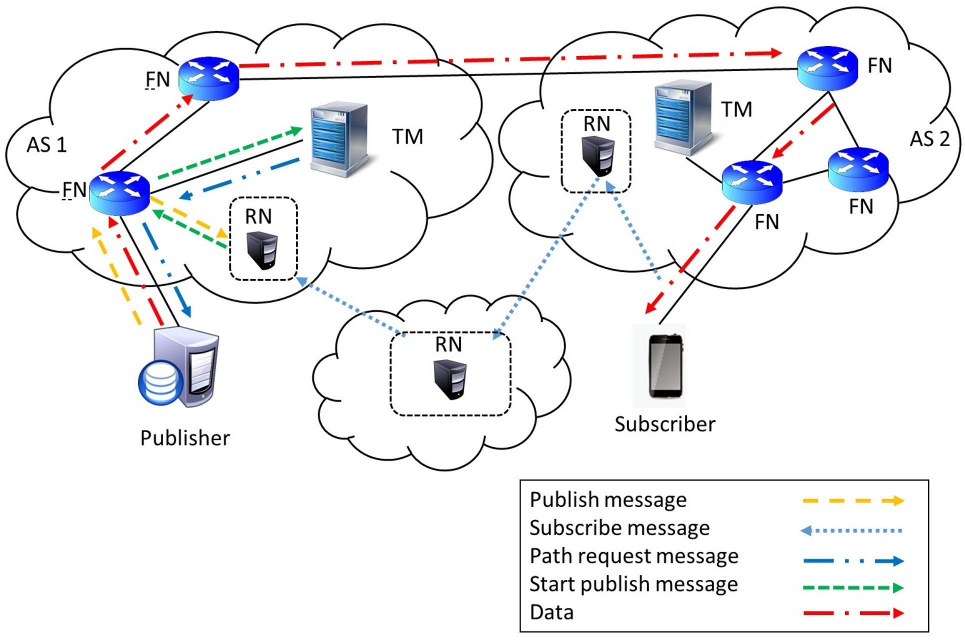


FIGURE 1.4: PURSUIT architecture.

1.2.4 CCN/NDN

Van Jacobson and et. al. initially elaborate the design and stricture of CCN, the pioneering fully-fledged ICN architecture, in the literature in 2009 [108]. Subsequently, The NDN project is funded in US, which is aiming to further improve the CCN architecture. By replacing the IP layer with the content chunks layer, NDN proposes a completely redesign of the Internet.

Name structure used in NDN is hierarchical similar to URLs. Generally, a content in NDN is consist of a set of small pieces of this content, namely chunk, which can be requested independently in sequence.

NDN communications rely on two types of packet, interest, and data. The interest packet is designed for requesting a target content and the data packet is for responding with the requested content. A typical NDN node consists of Content Store (CS) which temporarily caches the content routed to pass by, Pending Interest Table (PIT) that aggregates the requests so that returned data packet can be distributed to all consumer and Forwarding Information Base (FIB). The FIB of NDN is quite similar to an IP FIB, the improvement of FIB in NDN is that it allows one entry in FIB to hold multiple outgoing faces rather than a single one.

The communication over NDN is illustrated in Figure 1.5. When an NDN router receives an interest packet, the CS will be traversed by the content name carried by the interest packet in order to find whether the target chunk is cached there. If this chunk exists in the CS, it will be packaged into a data packet to be sent to the client directly. This is called a cache hit. Otherwise, PIT is checked in case there is a previous request asks for the same data. If there is a record in the PIT, the incoming face, which is the port of the incoming interest of this router, will be added to this entry and this interest packet will be drop. If there is no match in the PIT, the interest packet will be forwarded to upstream routers according to the FIB. When an NDN router receives a data packet, its CS will be traversed to find if there is the same data have been cache in this router before. The PIT record the incoming faces of each received interest packet by the content name. Therefore, If there is no match in the CS, according to the PIT record, the data will be forwarded to the faces which the interest was coming. The data will be cached in the CS while NDN routers are forwarding a data packet, and if the cache space is exhausted, a selected chunk will be evicted to make room for the incoming data.

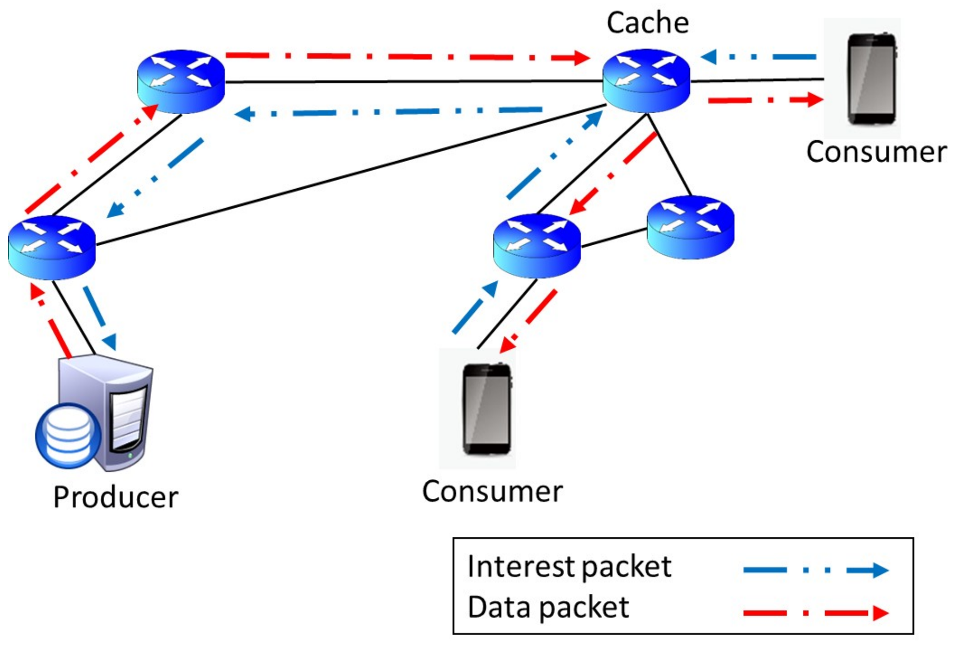


FIGURE 1.5: PURSUIT architecture.

1.3 In-network Caching in ICN

The efficiency of data transmission is further improved by adopting in-network caching, which has become a distinct research topic in the context of ICN [5] [6]. Named contents in ICN are able to temporarily stored in the intermediate network devices for the purpose of directly responding to the potential subsequence requests, which have the same content name. Precisely, in-network caching studies focus on the management of cached content in these network devices and aim to shorten the delivery distance between the content and users to reduce unnecessary network traffic [33–60]. Thus, the cache management strategies significantly affect the ICN network performance.

Cache management schemes can be classified into cache decision strategies and cache replacement strategies, as shown in Figure 1.6. The cache decision strategy focuses on choosing appropriate caching locations in the network for particular contents to reduce the cache redundancy and to improve the cache utilization, while a cache replacement strategy is necessary for each ICN router when the cache decision scheme is processed. The cache replacement strategy for ICN refers to the process that evicts a selected piece of content in the storage space to make room for incoming content.

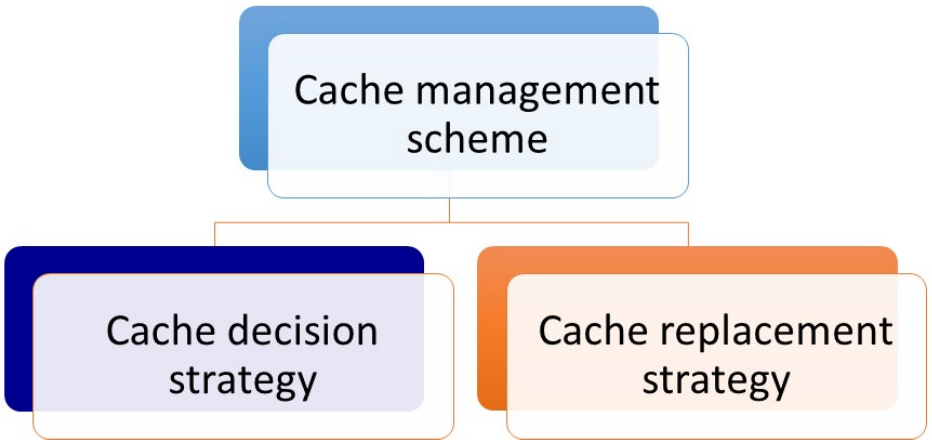


FIGURE 1.6: Cache management scheme.

A cache replacement strategy is necessary for each ICN router when the cache decision strategy is processed. For instance, Leave Copy Everywhere (LCE) and Least Recently Used (LRU) are adopted to work cooperatively as the default cache management strategy of NDN, as shown in Figure 1.7. Cache replacement is not a fresh issue raised by ICN. For memory management in computers, cache replacement algorithms are designed to improve memory page utilization. LRU, Least Frequently Used (LFU), First-In and First-Out (FIFO) and the Optimal replacement algorithm (OPT) [20] (also named Belady's algorithm) have been proposed to overcome the access speed difference between the slow auxiliary storage devices and the fast cache memory. Among these cache replacement algorithms, the OPT algorithm is the most inspiring one and has been proven to be theoretically optimal [21, 22]. The OPT algorithm evicts the item that will not be accessed for the longest period of time. However, this theoretically optimal algorithm is rarely implementable because it is difficult to predict when the cached items will be accessed in the future. In addition, the cache replacement issues for web caching scenarios are well studied [23, 24]. These studies can be classified as recency-based replacement (LRU, HLRU, Pyramidal Selection Scheme (PSS), etc.), frequency-based replacement (LFU, LFU-Aging, etc.), function-based replacement (Greedy Dual (GD)-Size, Taylor Series Prediction (TSP), etc.) and randomized-based replacement (RAND, etc.). Although these cache replacement strategies are not aiming for ICN, they are excellent references for designing the ICN cache strategy.

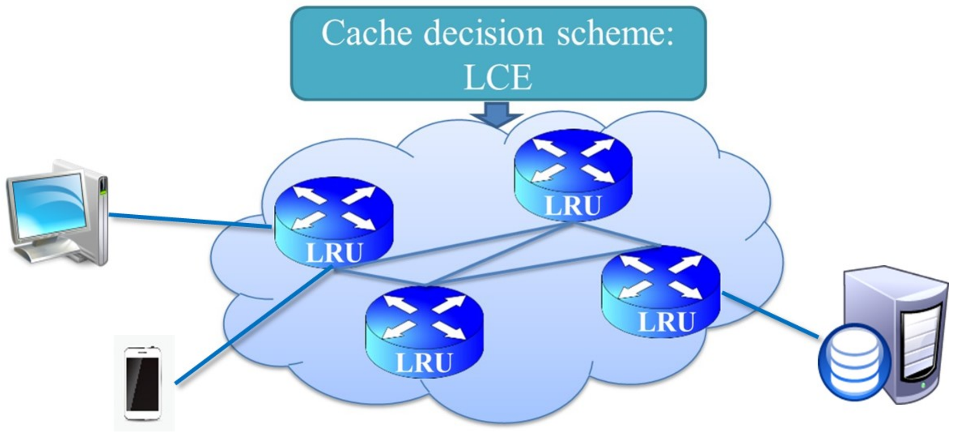


FIGURE 1.7: Default cache management strategy in NDN.

Web caching techniques have been studied for years and have become a practical approach for reducing bandwidth consumption in the current Internet. However, because of the new features in ICN structure, the well studied web cache techniques cannot be adopted to ICN seamlessly.

1. Cache transparency is different [49]. Web cache solutions are closed systems, and the cached content for one application can not be used by other application directly. And even for the same application, the cache copies of the same content cached in different routers are recognized as different content witch cannot be used efficiently. Nevertheless, ICN has a unique naming system, and the contents over ICN are forwarded, retrieved and cached only according to the unique content names. Therefore, ICN is able to offer an open transparency cache service to the application layer.

2. Unlike web cache that the cache locations are determined beforehand and close to the edge of the network, the cache in ICN is ubiquitous, in other words, in-network caching is supported by ICN [49]. Due to the unique naming mechanism in ICN, every indeterminate network devices are capable of temporarily cache the content passing by and able to respond the cached contents to the requests directly. In-network caching facilitates the contents delivery and shortens the average content delivery distance. Thus, the overall unnecessary network traffic among intermediate routers is reduced. As such, the management scheme of in-network caching will not only affect the cache efficiency but also influence the whole network performance significantly.

3. The cached object is in different granule size [49]. In the studies of traditional web caching, the unit of cache management usually is a file. However, within the ICN infrastructure, the unit of caching is a small segment of a file called a *chunk*. And this lead to the inadequacy in simply extending the file-based popularity conclusions to chunk-by-chunk

ICN in-network caching. Furthermore, for ICN cache, the request popularity for different chunks within a file is different, and this internal request popularity is still an open issue. In addition, *Independent Reference Assumption* (IRA) which is typically used in traditional file-oriented caching is no longer appropriate to ICN. Independent reference assumption refers that the request of a particular object is emerged independent and will not affect by other objects. In ICN, the requests for different chunk usually be generated relatively, i.e. these requests are required for the same file and generated in sequence. Thus the IRA may fail in this scenario.

1.4 Research Motivation and Aim

As one of the major features, in-network caching plays an important role in revealing the advantages of ICN. The performance of in-network caching scheme affects the ICN network performance dramatically. However, the current in-network caching studies are not efficient enough to manage the cache space in the ICN, and the cache efficiency in ICN still have room to be improved.

In-network caching specifically for streaming content delivery (e.g., video delivery, audio broadcasting and file distribution) over ICN architectures raises new challenges. Firstly, the cached object has a different granule size. This variation leads to inadequacy in simply extending the existing file-oriented caching research results to ICN caching studies, such as the popularity of cached objects and the request feature. Additionally, for streaming content delivery over ICN, the chunk request sequence for a file is well ordered rather than generated independently. Thus, the correlation in the occurrences of requests for chunks in the same file needs to be considered.

In this dissertation, I address the problem of cache management, which focuses on streaming content delivery over ICN. The research aim is to enhance cache utilization and reduce unnecessary network traffic by proposing cache management strategies including both cache decision algorithms and cache replacement algorithms. In addition, I also attempt to avoid additional management overhead, such as messages or control packets transmission, which increases the network burden to some extent.

1.5 Organization of the Dissertation

This dissertation is organized into 8 chapters. In this dissertation, 6 cache management strategies for streaming content delivery over ICN are proposed, including cache decision algorithms and cache replacement algorithms.

In chapter 1, the overview and research background of this research are initially introduced. Furthermore, the current research projects of NDN and the in-network caching in NDN is illustrated. According to the issues that the ICN architecture may face while delivering streaming content, the research motivation of this study is elaborated, as well as the research aim. In addition, the organization of this dissertation is presented at the end of this chapter.

In chapter 2, a content popularity based FLCD cache decision strategy is proposed, to rapidly cache the most popular contents at the edge of the network and cache the widely popular contents at the intermediate branch routers, in order to reduce the network storage redundancy and shorten the total retrieval time.

In chapter 3, DFLP-based cache management strategy including both cache decision and cache replacement strategy is proposed, which is designed specifically for streaming content delivery over NDN.

In chapter 4, two TTH based cache replacement strategies, TTH-LFU and TTH-LRU are proposed specifically for video streaming content delivery over NDN.

In chapter 5, by considering the characteristics of streaming content delivery, a cache replacement strategy named PPCSA is proposed which is an effort to enhance the cache performance of NDN. In this chapter, the PPCSA model and strategy are introduced where file-level and chunk-level popularity are initially proposed.

In chapter 6, TLP-TTH cache replacement strategy is proposed for video delivery over NDN, which consists of an evictee selection algorithm and a cache size allocation algorithm. The cache replacement strategy takes two levels of access probability into consideration and is customized for video streaming by carefully selecting evictees that have low access probability of being requested in the future.

In chapter 7, RXI based caching replacement strategy customized for streaming delivery over ICN is proposed. RXI is introduced to offer unified estimation criteria of possible future request probability for cached chunks. RXI adopts both file-level and chunk-level internal request probability and is estimated according to the dynamically varying request status at each ICN router.

In Chapter 8, I concluded the dissertation and stated the future work.

Chapter 2

FLCD Cache Decision Strategy

In this chapter, a content popularity based cache decision strategy named Fast Leave Copy Down (FLCD)[1] is proposed, to rapidly cache the most popular contents at the edge routers of the NDN and cache the widely popular contents at the intermediate branch routers, in order to reduce the network storage redundancy and shorten the total retrieval time.

2.1 Introduction

The main function of the the current Internet has gradually shifted from host to host communication to efficient content delivery. Thus future internet architectures, such as NDN, have drawn the extensive attention of researchers.

Two critical characteristics of NDN are in-network caching and name-based routing. With the feature of in-network caching, the contents are able to be cached along the path of the data transmission, thereby the subsequent requests can benefit from acquiring these content copies at a closer router without visiting the original sources. Due to the limitation of network storage capacity, the decision of which and where content should be cached will

[1]This work has been published in the proceedings of IEICE General Conference 2014, as No.6 paper listed in the list of publications.

affect network transmission performance tremendously. In the current NDN, the cache decision policy named LCE will keep the content on every node along the content transmission path. To improve the efficiency of In-network caching, new strategies have been proposed such as LCD [72] which will cache the content copies only on the node one hop away along the transmission path for each request. In this chapter, FLCD is proposed, which is a popularity based cache decision strategy to select the most popular content and rapidly cache them on the edge nodes of the NDN and cache the widely popular contents at the intermediate branch routers, in order to reduce the network storage redundancy and shorten the total retrieval time.

2.2 FLCD Cache Decision Strategy

The FLCD strategy is based on the popularity of contents, and it is designed to cache the ”hottest” contents near the requester and reduce the storage redundancy without costing much overhead of the network.

The proposed FLCD strategy stipulate that the most popular content is cached at the edge nodes, the widely popular contents are cached at the intermediate branch routers and other contents are cached at the next immediate node (one hop) to gain a shorter communication time of subsequent requests, as shown in Figure 2.1.

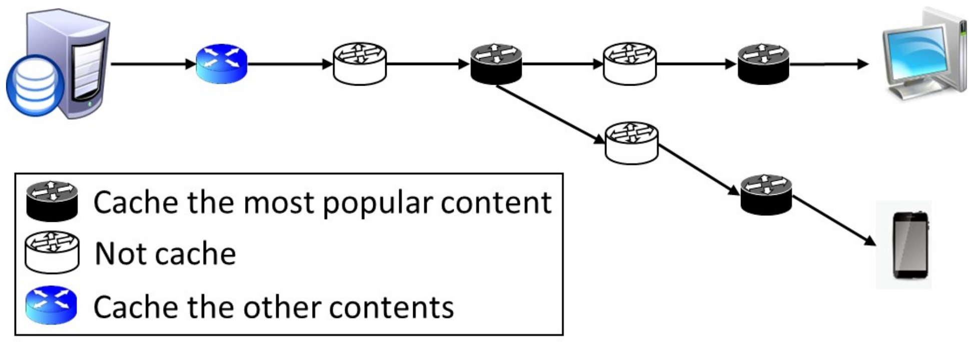


FIGURE 2.1: Cache decision strategies.

The FLCD strategy is composed of two algorithms, the *Decision algorithm* and the *Adjustment algorithm*. I record R_q which is the number of times that this content has been requested to determine the most popular content in this cache. And R_q will be removed when this content is replaced.

Based on the hop count of received interests packet I_{hp}, at cache hit node, *Decision algorithm* able to decide how many hops (D_{hp}) the content needs to be cached and data packet will bring this D_{hp} while being transferred. The I_{hp} is carried by interest packets and keep counting during delivery. If the requested content is the most popular content, the D_{hp} is set I_{hp} to make sure this content to be cached at the edge node, otherwise, the D_{hp} is set to 1 so that the requested content can be cached at the next router away from the source, as shown in Figure 2.1.

When the NDN node receives a Data packet, the D_{hp} will be checked and the data will be cached in CS if the D_{hp} equals to 1, otherwise, the D_{hp} will be decreased by 1 and then the packet will be forwarded to the next node, as shown in Algorithm 2.1.

Algorithm 2.1 Decision Algorithm

```
Require: $I_{hp}$
1: if requested content $C_i$ exsist in CS then
2:     Check the $R_q$ of $C_i$;
3:     $R_q \leftarrow R_q + 1$;
4:     if $R_q$ is the most largest one then
5:         $D_{hp} \leftarrow I_{hp}$;
6:     else
7:         $D_{hp} \leftarrow 1$;
8:     end if
9: end if
```

Adjustment algorithm is to adjust the I_{hp} carried by data packets while it is being forwarded. By considering that the interest packets may be aggregated by NDN nodes, and the hops to the edge node through different faces may be different, the I_{hp} of interest packets form different faces are recorded in the PIT entry. Accordingly, the Adjustment algorithm is able to modify the D_{hp} to the correct hops to make sure this content can be cached at the edge. Besides, when a node finds that the received content was requested by multiple faces and the number of these faces is the largest one, this content will also be cached at this branch node as shown in Algorithm 2.2.

Algorithm 2.2 Adjustment Algorithm

```
Require: $D_{hp}$, face set $F\{F_1, F_2, ..., F_m\}$, $I_{hp}$
 1: if received data packet $C_i$ has request record(s) in PIT then
 2:     if $C_i$ was requested by the largest numbers of faces then
 3:         Cache $C_i$ at this node;
 4:     end if
 5:     for each request face $F_i$ do
 6:         if $D_{hp} \neq I_{hp}$ then
 7:             $D_{hp} \leftarrow I_{hp}$;
 8:         end if
 9:         Forward data through face $F_i$ with $D_{hp}$;
10:     end for
11: end if
```

When the first few requests are sent to the network, the R_q makes little difference with others. Hence it is not easy to distinguish the most popular content from others. Therefore, starting-threshold STN is introduced which is set based on Zipf-distribution. If the numbers of CS entry reached the STN, the FLCD will be triggered, otherwise, content will be cached on next hop. As shown in equation 2.1, N is the cache size of CS and α refers to the Zipf rank exponent.

$$STN = \frac{\frac{1}{2^{\alpha}}}{\sum_{i=1}^{N} \frac{1}{i^{\alpha}}} \times N \tag{2.1}$$

2.3 Simulation and Results

FLCD is implemented in ndnSIM [71] simulator. To gain analytical insights, I simulated the FLCD, LCD and LCE cache decision strategies independently. The cache replacement strategies of this simulation are LRU, LFU, and Random.

4-levels complete binary tree is selected as the topology of the simulation, and the leaf nodes refer to the edge of the network. The Producer is an abstraction of the multiple original content sources which are connected to the edge of the network, and other 7 consumers are connected to other leaf nodes. 10000 unique contents (1KB for each) are used which are requested with the popularity that follows the Zipf-distribution. The cache size of each NDN router is set to 100K. This simulation does not stop requesting unless 10000 contents have been received by all consumers.

Two parameters: total retrieval time and average retrieval distance are used to measure the performance of the proposed strategy for this simulation.

The total retrieval time illustrates the time when all the consumers successfully receive every content. In Figure 2.2, FLCD have the smallest total retrieval time with different cache replacement strategies compared to LCD and LCE.

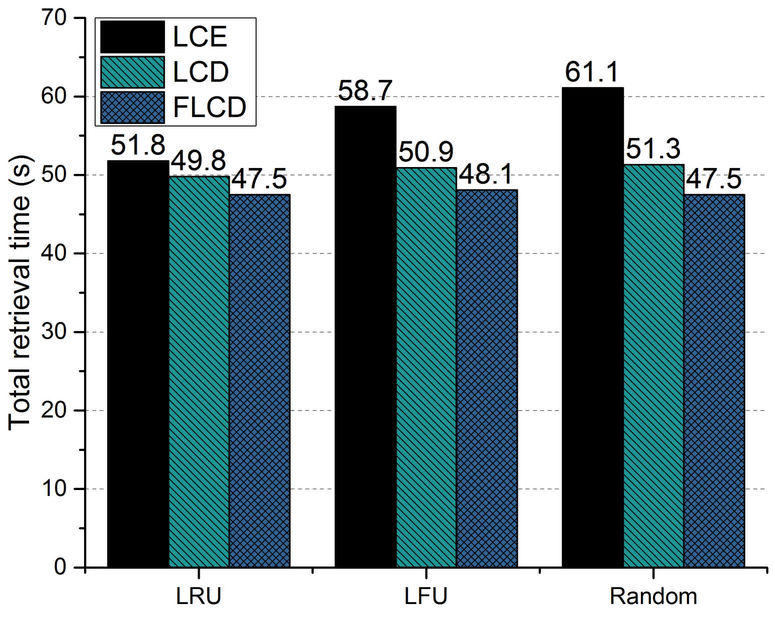


FIGURE 2.2: Total retrieval time.

The average retrieval distance is the combined number of hops for Interest and Data for one request. In Figure 2.3, as expected, the FLCD reduces the average retrieval distance evidently than other two strategies.

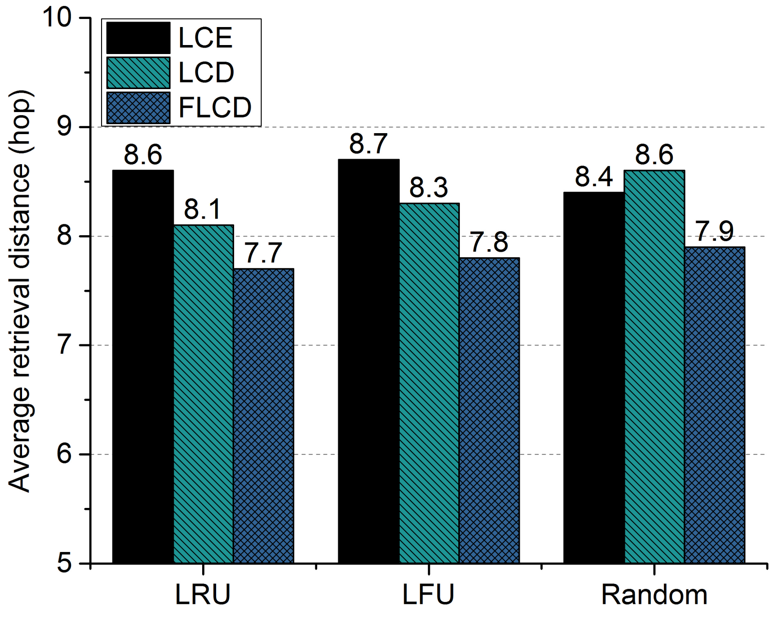


FIGURE 2.3: Average retrieval distance.

2.4 Summery

In this chapter, a popularity based cache decision strategy FLCD is proposed for the cache management issue in ICN networks. Simulation results show that, compared with LCD and LCE, FLCD achieves better performance, in terms of shorter retrieval time and less retrieval distance.

Chapter 3

DFLP Cache Replacement Strategy

In this chapter, a Dynamic File Level Popularity (DFLP)[1] based cache management strategy is introduced to improve the performance of streaming content transmission over NDN. With the feature of streaming content delivery, I take the file-level popularity of streaming files into consideration instead of the chunk-by-chunk popularity of individual streaming chunks. The results show that DFLP can reduce the total retrieval time and average retrieval distance, meanwhile increase the average cache hit ratio.

3.1 Introduction

NDN cache management strategy is composed of cache decision strategy which is for deciding whether to cache at the time of a data packet arrival and cache replacement strategy which is used to choose an appropriate content in CS to be replaced to make room for new incoming content.

NDN preforms variedly according to different cache management strategies. Thus researchers are attracted to finding more effective cache management strategies in order to improve network performance. Xiaoyan Hu, et al. proposed a Not So Cooperative Caching strategy (NSCC) [73] which is

[1]This work has been published in the proceedings of IEICE General Conference 2015, as the No. 5 paper listed in the list of publications.

an extension study of their previous work named Cooperative Caching strategy, introduced a cache decision policy based on access cost of selfish nodes. These selfish nodes would cooperate in caching and sharing content if and only if each of the benefits. The storage redundancy is dramatically reduced nevertheless the scalability of NSCC is under discussion. Two cache replacement algorithms LGD-size and Lmix [74] were proposed for layered video content streaming in CCN, by Junghwan Lee and et al. Their experiment results showed that the proposed policies maximum double the cache hit ratio for each NDN node compared with the FIFO, LFU, and LRU.

3.2 DFLP Based Cache Management Strategy

DFLP is a complete cache management strategy includes a DFLP cache decision strategy working corporately with the DFLP cache replacement strategy. In this study, I assumed that a single streaming file $f_i = \{C_{i,1}, C_{i,2}, \ldots, C_{i,n}\}$ ($i \in m$, which m is the number of streaming files) consists of n independent chunks $C_{i,j}$ which will be requested in a row by viewers according to the unique sequence number j ($j \in n$), and the sequence number j are ordered by streaming request sequence. The file-level popularity Req_i of different streaming files is measured instead of chunk-by-chunk popularity $Req_{i,j}$, and according to the recorded file-level popularity Req_i, I carry out our cache decision and cache replacement strategies.

3.2.1 DFLP Cache Decision Strategy

The DFLP cache decision can be divided into two steps: monitoring the file-level popularity Req_i while forwarding interest packets, and making cache decision while receiving data packets according to the Req_i and file occupancy rate of CS Rcs_i. Rcs_i refers to the storage space occupancy rate by this streaming file f_i in the content store. Req_i and Rcs_i are defined as follows:

$$Req_i = \frac{\sum_{j=1}^{n} Rec_{i,j}}{\sum_{i=1}^{m} \sum_{j=1}^{n} Rec_{i,j}} \tag{3.1}$$

$Rec_{i,j} \in \{0, 1, 2, ...\}$ refers to the number of requests for chunk $C_{i,j}$ which is measured in a single time unit while NDN node forwarding interest packets. Thus, according to formula 3.1, request rate Req_i able to demonstrate the dynamic request popularity of streaming file f_i under different communication stages.

$$Rec_i = \frac{\sum_{j=1}^{n} Rcs_{i,j}}{K} \tag{3.2}$$

$Rcs_{i,j}$ indicates that the real time cache occupancy rate for streaming file f_i in CS. In formula 3.2, $Rcs_{i,j} \in \{0, 1\}$ addresses whether chunk $C_{i,j}$ is cached in this CS and K is the cache size of this node.

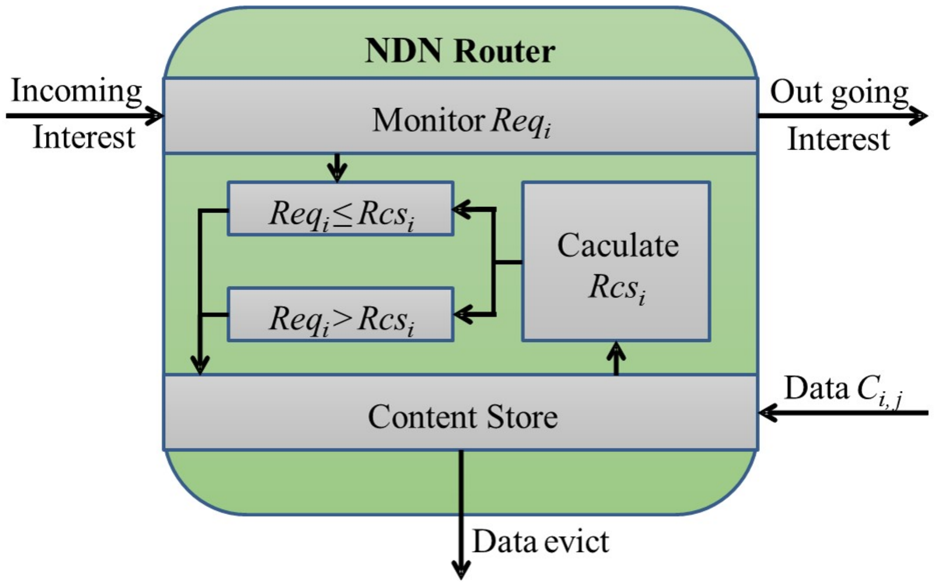


FIGURE 3.1: DFLP cache decision strategy.

The cache decision is made according to Req_i and Rcs_i. When an NDN node receives a data packet of chunk $C_{i,j}$, I compare the file request rate Req_i

and the cache occupancy rate Rcs_i. If Rcs_i is smaller, that is to say CS still have enough room for file f_i and content $C_{i,j}$ is decided to be cached, as shows in Figure 3.1. , otherwise, do not cache.

3.2.2 DFLP Cache Replacement Strategy

With request rate and CS occupancy rate changes dynamically, the DFLP cache replacement varies by different scenarios.

When NDN node receives a data packet of content $C_{i,j}$ and CS is full, I compare the file request rate Req_i and the cache occupancy rate Rcs_i. if $Req_i \leq Rcs_i$ in other words, file f_i has already occupy more storage spaces than it needed, thus a content $C_{i,q}$ in the CS is replaced which not only has the same file name with the new coming content $C_{i,j}$ but also is the less frequency used content. On the other hand, if $Req_i > Rcs_i$ which means more contents of file f_i need to be cached to meet the high request rate, then I choose content $C_{p,q}$ in the CS to remove, and content $C_{p,q}$ is the less frequency used one with different file name of new coming content $C_{i,j}$.

As shown in Figure 3.2, e.g. when node receives content $C_{1,2}$, if $Req_1 \leq Rcs_1$, content $C_{1,4}$ will be removed. On the contrary, if $Req_1 > Rcs_1$, content $C_{2,6}$ will be replaced according to DFLP cache replacement strategy.

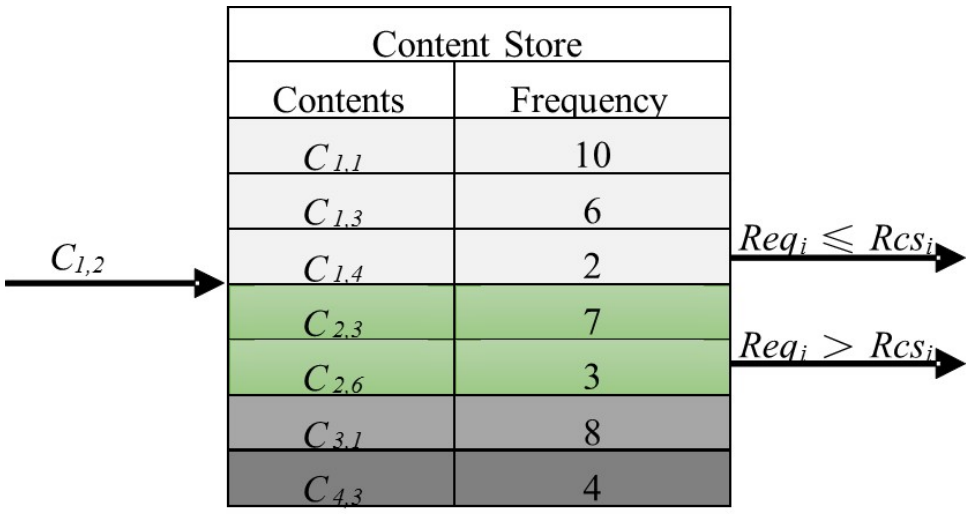


FIGURE 3.2: Sample of DFLP cache replacement strategy.

3.3 Simulation and Results

I implemented our scheme in ndnSIM [71] simulator. To gain analytical insights, I compare our proposed scheme DFLP with NDN cache decision strategy LCE and two cache replacement strategies LRU and LFU independently under a hybrid topology which composed of a core network and access networks. Abilene [75] is chosen for the core network and for the access networks I used 3-levels complete binary tree. I implemented 25 different streaming file producers, each of them offered a unique streaming file, each streaming file has 800 chunks, whose size is 800KB, and the total chunk number is 20,000. I assume that 100 consumers request for different streaming files. The access frequency of the files follows the Zipf-distribution ($\alpha = 0.8$). Besides, the consumers will request the streaming chunks in the order of the sequence number j.

Total retrieval time is the time when all consumers have successfully received every requested chunk. As shown in Figure 3.3, our proposal shortens the total retrieval time.

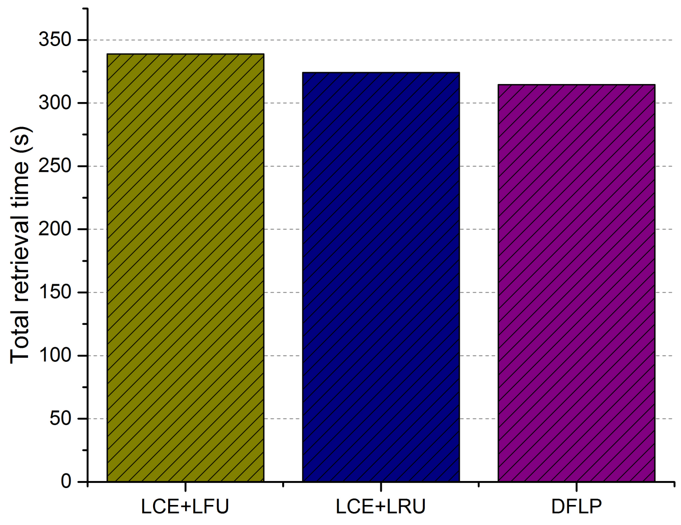


FIGURE 3.3: Total retrieval time.

Average retrieval distance is the combined number of average hops for interest and data delivery. In Figure 3.4, DFLP reduce the average retrieval distance compared to the other two cache management strategies.

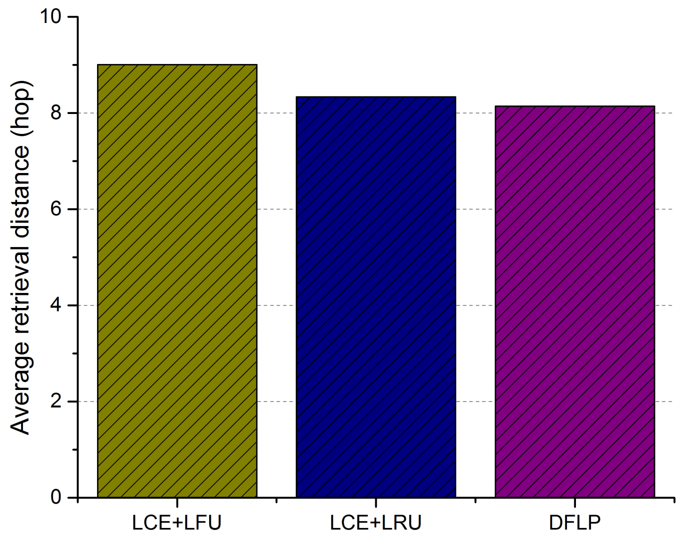


FIGURE 3.4: Average retrieval distance.

As is clearly shown in Figure 3.5, the improvement of cache hit ratio performed by DFLP can be up to nearly 20%.

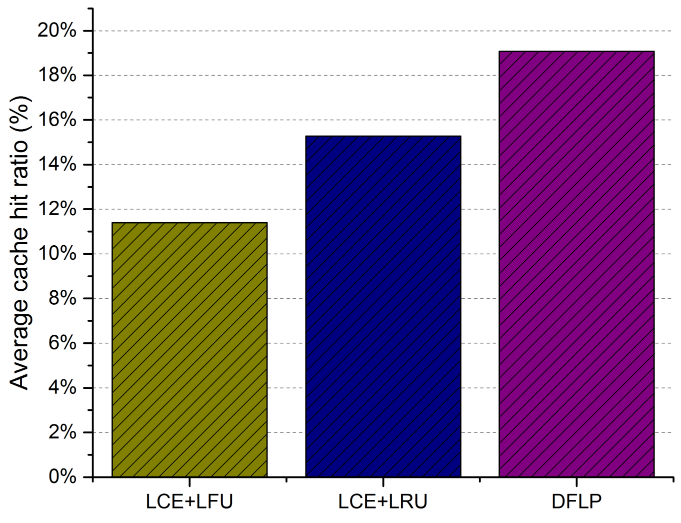


FIGURE 3.5: Average cache hit ratio.

3.4 Summery

In this chapter, I proposed DFLP based cache management strategy for streaming content delivery over NDN which composed with both cache decision strategy and cache replacement strategy. And the simulation results showed that compared to the default NDN cache management schemes LCE+LRU and LCE+LFU, the proposal improved the network performance by reducing the total retrieval time and increasing the cache hit ratio.

Chapter 4

TTH Cache Replacement Strategy

In this chapter, two Time to Hold (TTH)[1] based cache replacement strategies TTH-LFU and TTH-LRU are proposed, which not only consider the features of video delivery over NDN but also take the future popularity into account. The research goal is to reduce the cache storage redundancy and to improve the network performance while costing quite limited overhead. In the simulation, the strategies are compared against two widely accepted cache replacement strategies LRU and LFU on 4 different hybrid topologies with variable cache sizes. The results validate that for every scenario, the proposed strategies significantly improve the cache hit ratio and shorten the total retrieval time as well as the average retrieval distance, and lighten the burden imposed on the original sources. Furthermore, The proposals can decrease the excessive network traffic, which avoids the unnecessary data delivery among routers and reduces the transmission energy consumption 20% on average from LRU and LFU. Besides, the impact of several factors is evaluated to demonstrate that how other factors affect the performance of our proposed cache replacement strategies.

[1]This work has been published in IEICE Technical report. CS, 2014, as the No. 4 paper listed in the list of publications.

4.1 Introduction

In-network caching, one of the characteristics of NDN is able to avoid the redundant network transmission. An appropriate cache management strategy enhances the NDN network performance. Cache management strategy is composed of cache decision strategy which decides where to cache these contents and cache replacement strategy for removing a carefully selected content in the cache to make room for the new coming one.

There is clearly a large number of works related to the performance of caching in NDN. Yanhua Li, et al. [76] developed a holistic model to analyze the optimal strategy for providing the in-network storage capability and presented an optimal strategy to provide the storage capability that optimizes the overall network performance and cost. Based on off-path caching, Martin Draxler and Holger Karl [77] proposed alternative strategies which try to avoid redundant caching of contents to use cache space more efficiently. Yonggong Wang, et al. [78] investigated an analytical method to find an optimal solution in deciding which CCN node should be equipped with the cache function and how many contents should be cached. By using this method, they are capable to analyze what and how the factors, such as topology, network size, content popularity characteristics and different cache replacement policies affect the network performance.

Since NDN is more suitable to deliver a large number of contents, in particular, video streams, some researchers are devoted to the studies of video transmission experiences and application design through NDN. Suphakit Awiphan, et al. [79] proposed a framework to study the performance of video streaming over NDN, and showed that overlay delivery path and size of data chunks affect the streaming quality. Meanwhile, Andrea Detti, et al. [80] presented a P2P application for live streaming delivered over NDN.

In this chapter, I focus on cache replacement strategies for video delivery over NDN and propose two cache replacement strategies TTH-LFU and TTH-LRU. The proposals not only consider the current request popularity of contents but also take future request popularity into account. TTH replacement strategies will "Hold" the contents of the CS that will likely be requested in the coming seconds to improve the network transmission capacity.

4.2 TTH Cache Replacement Strategy

4.2.1 Assumptions

NDN is a content-oriented network architecture in which packet routing is based on unique content names, and the communication is driven by consumers. A consumer asks for a specific content by broadcasting interest packets to every available face according to this unique content name. NDN able to use hierarchical naming structure (e.g. */www.youtube.com/2014/video03/No004*). Any node receiving the interest packets and having data with the same name can respond with a data packet. And the files such as video streams can be transferred segment by segment.

Therefore, I assume that the name of requested video contents in NDN is composed of a Pre-Name (PN) that identify this specific video file and a Sequence Number (SN) which able to distinguish different content segments in this video. And the SNs are ordered based on video playback time.

4.2.2 TTH Model

Time to Hold (tth) is the estimated time point until which the already cached chunks are expected to be requested in the future. Since I assume that consumers will request the video chunks in a row according to the sequence number, I deduce that the subsequent chunks will have a high probability to be requested in the next time interval. For instance, as shown in 4.1, if an NDN router received an interest packet which requests for *video01/0003*, the subsequent contents such as *video01/0004*, *video01/0005*, etc. will highly be requested later. Therefore, if any of these subsequent contents have been cached in CS, they will have a higher chance to be requested than other contents. And our aim is to "hold" these contents in the CS for a short time span, to make sure that these contents will not easily be replaced by the cache replacement strategy. And the end time of this time span is tth as I defined.

However, it is obvious that not all subsequent chunks need to be held in the CS if the sequence number is much larger than the current request one (e.g. *Video01/0999*). Thus, I bring out Update Window Size (UWS) to manage how many subsequent contents should be held. E.g. assume that current request is for *video01/0003* and UWS equal to 10, only the contents whose sequence numbers are between $/0004$ and $/0013$ will be kept if they have been cached, as shown in Figure 4.1.

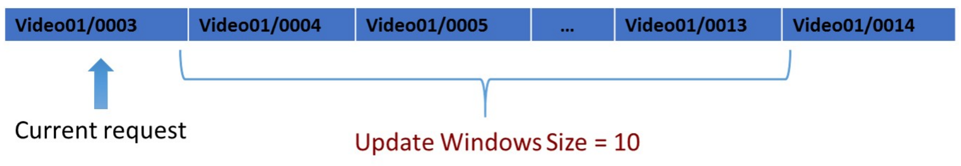


FIGURE 4.1: Update Window Size (UWS).

Every tth of contents are recorded in the CS. When a content initially be cached in the CS, the tth of this content is the cached time. And all tth should be updated accurately when NDN router receives an interest packet. And the contents of which tth will be updated must under the following constraints:

1. The content have the same Pre-Name with the received Interest packet.
2. Sequence Numbers is bigger than the received interest packet.

3. The gap of Sequence Numbers between two contents is below Update Window Size.

The tth is calculated by sequence number gap between the just received interest packet and the cached contents. As shown in equation 4.1, t_0 is the time that interest packet is received, SN_i is the sequence number of $content_i$ that stored in the CS and $SN_{Interest}$ is the sequence number which is carried by the just received interest packet. The average playback time for a single video chunk is present by Δt and C for tiny interval to keep this content in CS a little bit longer.

$$TTH = t_0 + (SN_i - SN_{Interest}) \times \Delta t + C \tag{4.1}$$

4.2.3 tth Update

When NDN router receives an interest packet, all the tth in the CS will be checked in case for update. Consequently, the contents which I aim to hold will have a larger tth and update more frequently than others which may unpopular or have low probability to be requested later. The procedure is clearly presented in Algorithm 4.1, where $PN_{Interest}$ represents the Pre-Name of the received interest and the PN_i is for $Content_i$ that cached in the CS.

4.2.4 TTH-LFU Cache Replacement Strategy

TTH-LFU, one of our proposed cache replacement strategy, is not only considering the content-level request frequency but also take file-level frequency into account. This cache replacement strategy will remove the content which both least frequently used and have less chance to be ordered in the coming time interval. According to the long-tail theory, the request

Algorithm 4.1 tth Update

```
Require: $PN_{Interest}$, $SN_{Interest}$, $t_0$, $UWS$;
1: Receive an interest packet;
2: for $i <= CacheSize$ do
3:     if $PN_i = PN_{Interest}$ then
4:         if $SN_i > SN_{Interest}$ and $SN_i - SN_{Interest} <= UWS$ then
5:             $TTH = t_0 + (SN_i - SN_{Interest}) \times \Delta t + C$;
6:         end if
7:     end if
8: end for
```

frequency of last few unpopular contents makes limited difference. Due to the consideration all above, I first choose a group of least frequently used contents as replace candidates which are kept in a temporary buffer named Candidate Buffer ($CB[n]$), and then compare tth_j, the tth of $content_j$ in this $CB[n]$, to choose the earliest tth one to remove it. Therefore, the popular contents and the contents which will highly be requested later will continue to be cached. As shown in Algorithm 4.2, $Content_r$ is the aimed content that I want to replace whose tth is presented by tth_r.

4.2.5 TTH-LRU Cache Replacement Strategy

If I extend the size of candidate buffer to the CS cache size, another cache replacement strategy named TTH-LRU is proposed, which only based on content tth. The processing is presented in algorithm 4.3.

4.3 Simulation and Results

I implemented our strategies in ndnSIM [71] simulator. To gain analytical insights, I simulated the LRU, LFU, TTH-LFU and TTH-LRU replacement

Algorithm 4.2 TTH-LFU

```
1: if CS is full then
2:     Initialize $CB[n]$;
3:     for $i <= CacheSize$ do
4:         $CB[n] \leftarrow n\ least\ frequently\ used\ contents\ in\ CS$;
5:     end for
6:     $Content_r \leftarrow CB[0]$;
7:     for $j <= n$ do
8:         if $tth_j < tth_r$ then
9:             $Content_r \leftarrow CB[j]$;
10:        end if
11:    end for
12:    Remove $Content_r$;
13: end if
```

Algorithm 4.3 TTH-LRU

```
1: if CS is full then
2:     for $i <= CacheSize$ do
3:         if $tth_i < tth_r$ then
4:             $Content_r \leftarrow Content_i$;
5:         end if
6:     end for
7:     Remove $Content_r$;
8: end if
```

strategies independently under different conditions. LCE, the default cache decision strategy, had been used in our simulation.

4.3.1 Simulation Configuration

The simulation network topologies are 4 hybrid topologies which are composed of core networks and access networks. For the core networks, Tree, CERNET, Abilene and GEANT [81–83] have been chosen and for the access networks I used 3-levels complete binary tree, as shown in Figure 4.2.

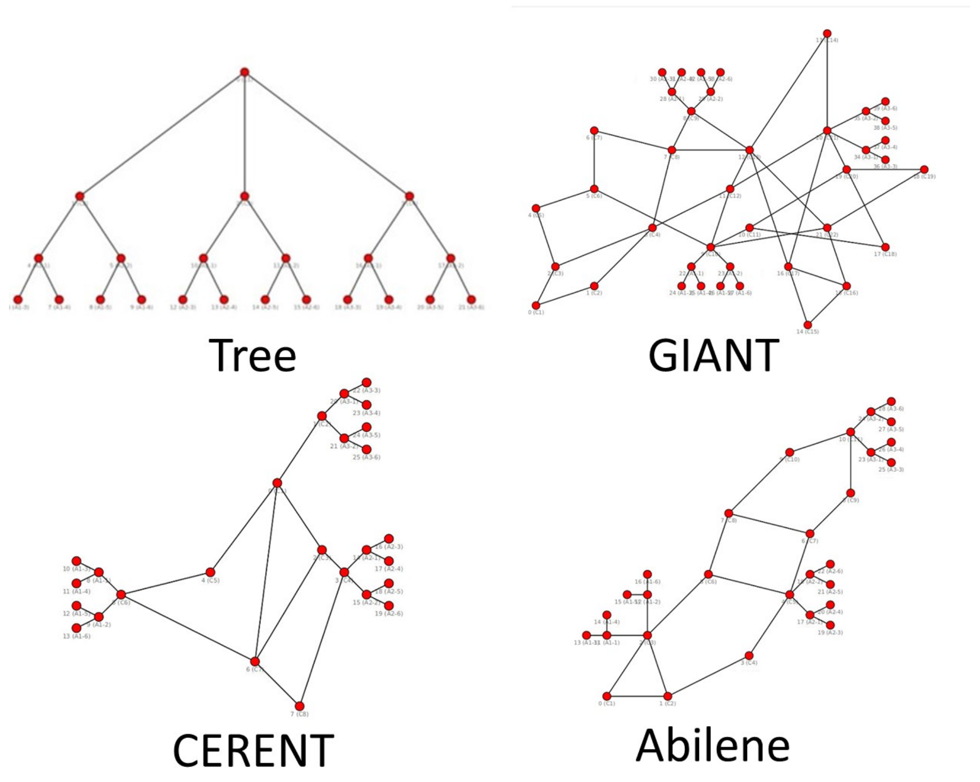


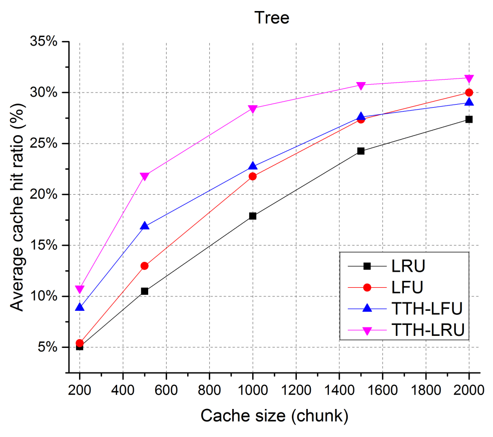
FIGURE 4.2: Simulation topologies.

To evaluate the performance similar to the real network circumstance, producers are connected to the edges of the access networks. I implemented 25 different producers, each of them offered a unique video and connected to 3 leaf nodes of the access binary tree on different branches. Each video has 800 segments, whose size is 800KB, and the total chunk number is 20,000. Furthermore, studies have shown that the popularity of video files follows Zipf distribution[84], thus I assume that 100 consumers request for different Pre-Name (different video files) follows the Zipf distribution ($\alpha = 1.7$), and the time interval of the request for different video files follows the exponential distribution. Referring to a single video, the consumers will request the video contents in the order of the SNs, and the time interval of request for these contents follows the uniform distribution with the mean is 1s. This simulation did not stop requesting unless 20,000 contents have been received successfully by the suitable consumers. I compare the effectiveness of our proposal with two cache replacement strategies LRU and LFU across a range of cache sizes, from 200 items to 2000 items for each NDN node.

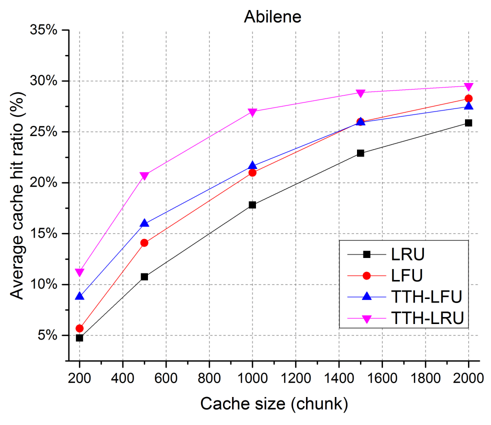
4.3.2 Results and Discussion

Average cache hit rate is often used as the main performance metric to measure the caching performance in the literature. In addition, I also take total retrieval time which illustrates the time for all consumer have successfully received requested contents, and average retrieval distance which is the combined number of hops for interest and data packets during one pair of request and respond to indicate the performance of our proposal. Besides, I calculated the network traffic as well on the different scenario to elaborate the reduction of unnecessary communication among routers.

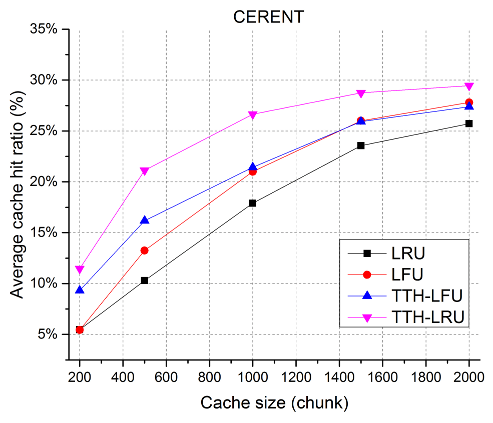
As shown in Figure 4.3, under variable cache size and topologies, The TTH-LRU always reaches the highest cache hit ratio compare to LRU and LFU. The improvement of TTH-LFU is inconspicuous on this request model, especially when the cache size is big enough.



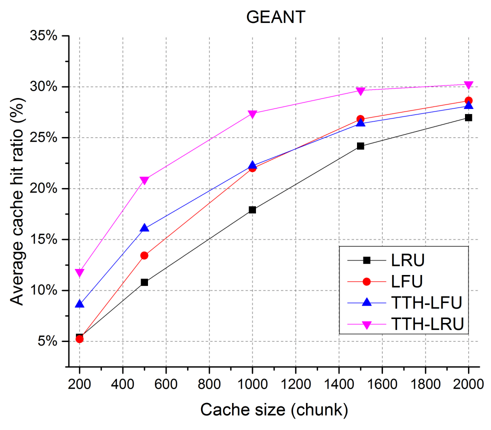
(a)



(b)



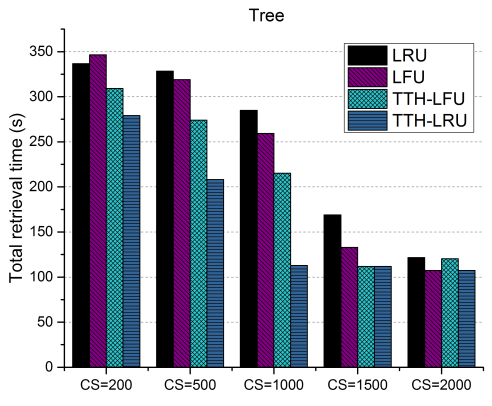
(c)



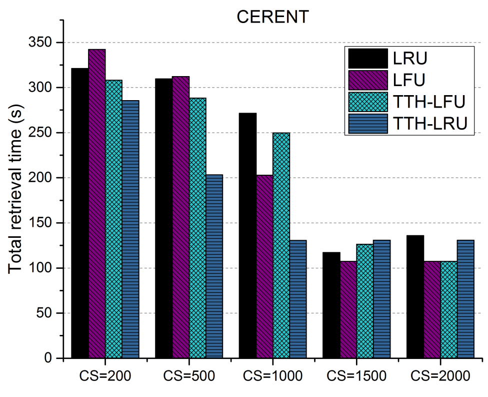
(d)

FIGURE 4.3: Average cache hit ratio. (**a**) Tree; (**b**) Abilene; (**c**) CERENT; (**d**) GERENT.

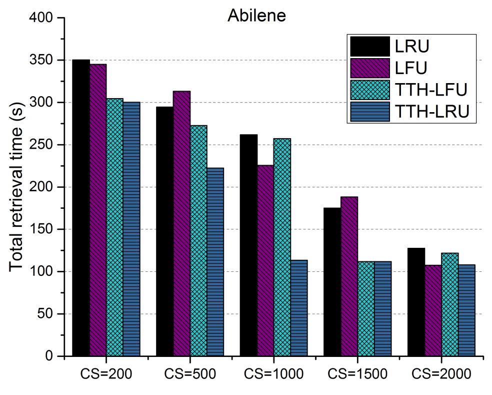
Total retrieval time, illustrate the total download time for all consumer have successfully received the segments of the video is shown In Figure 4.4. In this figure, both TTH-LFU and TTH-LRU reduced the total finish time with different cache size and topologies compared to LRU and LFU. Namely, our proposal is able to reduce the video buffer time, especially when the cache size is limited.



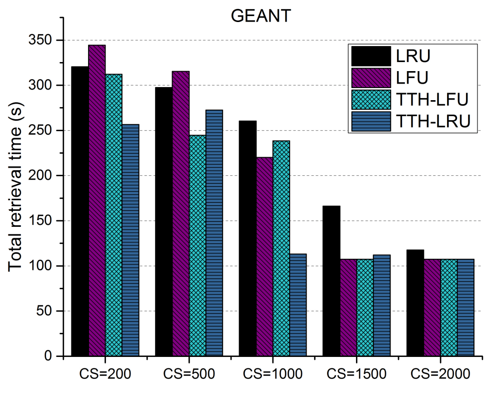
(a)



(b)



(c)



(d)

FIGURE 4.4: Total retrieval time. (**a**) Tree; (**b**) Abilene; (**c**) CERENT; (**d**) GERENT.

Average retrieval distance is another significant metric which indicates the average hops for data delivery over NDN network. In Figure 4.5, as expected, the TTH-LRU reduces the average retrieval distance evidently than other three cache replacement strategies.

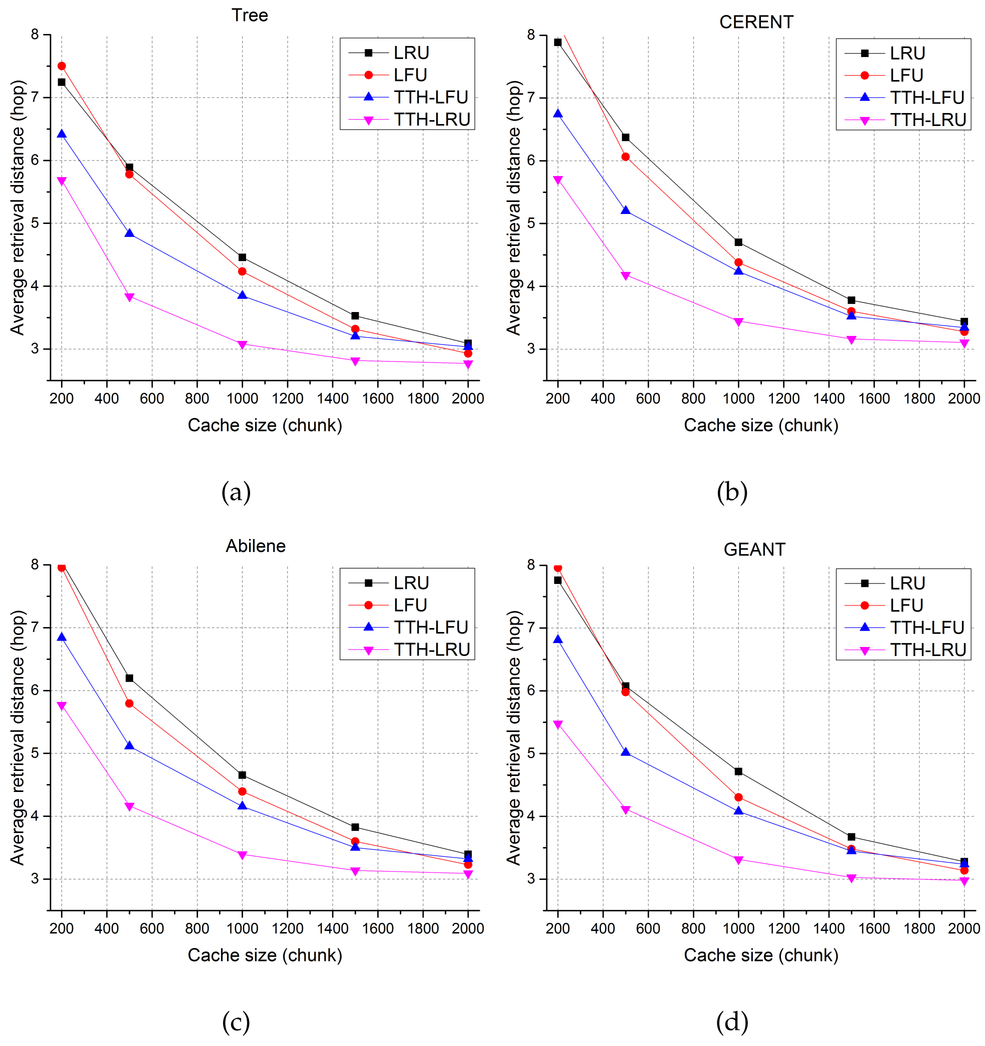


FIGURE 4.5: Average retrieval distance (hop). (**a**) Tree; (**b**) Abilene; (**c**) CERENT; (**d**) GERENT.

To expound the reduction of unnecessary transmission among NDN routers, I measured the total traffic of the network. I found that with the rising of the cache hit ratio, the utilization of cached content was improved, and hence popular contents do not have to be delivered among routers regularly. Thus, the total traffic of the network could be reduced. Table 4.1 report that compared to LRU, TTH reduced video transmission 22.6% on average under every scenario, and 20.6% from LFU. That is to say, TTH strategy can reduce around 20% of the transmission energy consumption compare with LRU and LFU.

TABLE 4.1: Network Traffic.

	Cache Size	TTH v.s. LRU	Average	TTH v.s. LFU	Average
Tree	200	16.23%	21.84%	16.26%	21.19%
	500	34.88%		38.51%	
	1000	29.74%		31.15%	
	1500	19.06%		15.58%	
	2000	9.25%		4.44%	
CERENT	200	21.04%	22.54%	21.56%	20.57%
	500	35.44%		36.67%	
	1000	30.98%		27.38%	
	1500	16.42%		13.00%	
	2000	8.79%		4.25%	
Abilene	200	26.31%	23.60%	22.94%	20.77%
	500	36.64%		33.59%	
	1000	28.88%		29.81%	
	1500	17.99%		13.86%	
	2000	8.19%		3.63%	
GEANT	200	19.49%	22.56%	19.51%	19.88%
	500	36.81%		34.34%	
	1000	31.35%		27.71%	
	1500	17.05%		13.73%	
	2000	8.17%		4.09%	

4.3.3 Factors Affect TTH Performance

To highlight the aspects that probably affect the performance of our proposed cache replacement policies, another group of simulations had been carried out. I compared the performance of TTH-LRU over different Update Window Size and TTH-LFU with different Candidate Buffer Size. I observe that the Update Window Size greatly influences the performance of TTH-LRU. And for TTH-LFU, to find a best Update Window Size could not reduce the average retrieval distance obviously when Candidate Buffer Size is quite limited. On the other hand, when I increased the Candidate Buffer Size of TTH-LFU from 20 items to 50, as shown in Figure 4.6, the performance of TTH-LFU (50) improved markedly.

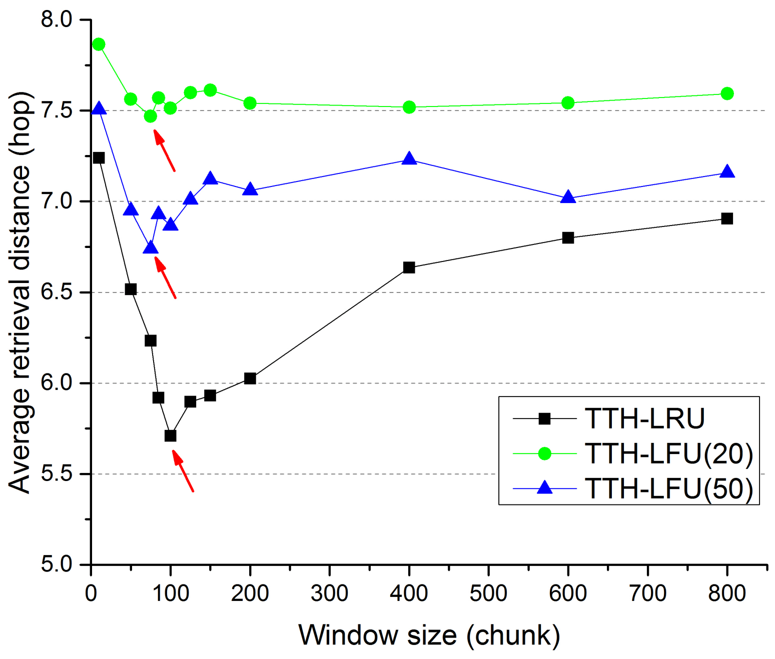


FIGURE 4.6: Average retrieval distance under different UWS.

4.3.4 Best UWS

Since that an appropriate UWS will significantly improve the performance of TTH-LRU, I aim to find some corresponding factors that will affect the Best Update Window Size.

Topology: I carried out another group of simulations on 4 different topologies, with the same request model configuration. The result shows that topology does not affect the best update window size, as is shown in Figure 4.7.

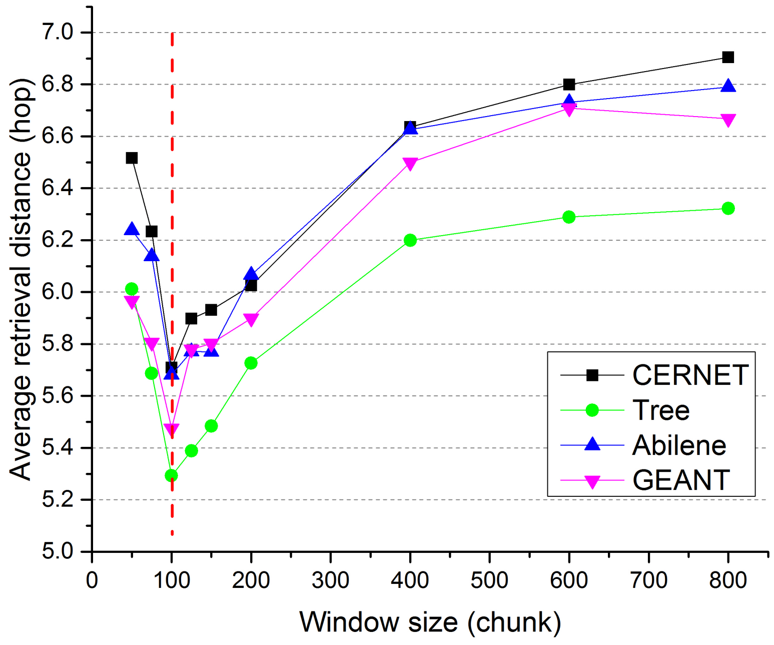


FIGURE 4.7: Best UWS on different topologies.

Cache Size: I simulated our proposal with different cache sizes and the best window sizes are distinct. Figure 4.8 illustrate that cache size do change the best update window size, and directly proportional to cache size.

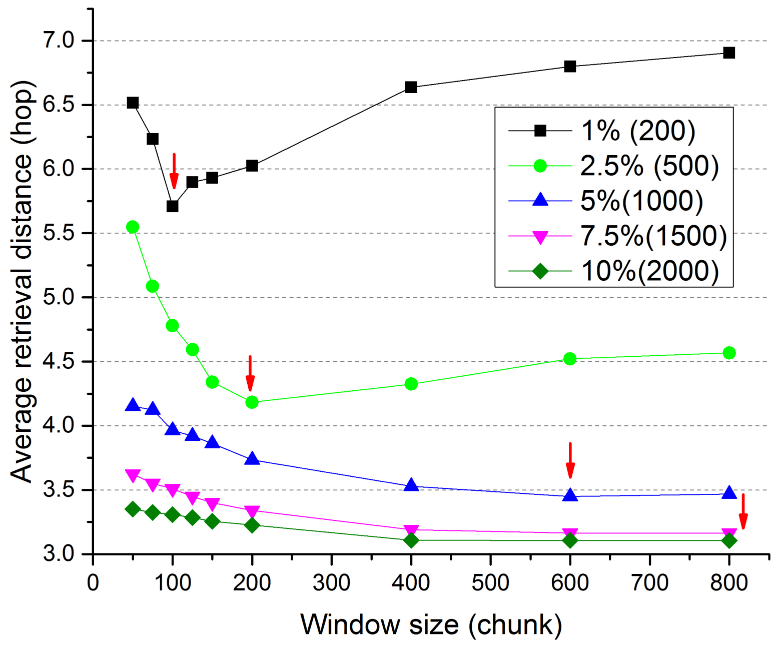


FIGURE 4.8: Best UWS on different Cache Size.

4.4 Summery

In this study, two cache replacement strategies TTH-LFU and TTH-LRU are proposed for improving the performance of video delivery over NDN. The results show that our strategies can shorten the total retrieval time and the average retrieval distance, and increase the cache hit ratio compared with LRU and LFU. In addition, TTH reduces the unnecessary network traffic that can be interpreted as the decrease in the transmission energy consumption around 20% on average. Besides, I evaluated the impact of several factors such as UWS and replace Candidate Buffer Size to demonstrate that how these factors affect the performance of our proposed schemes. And then I carried out another group of simulations to address the influence of best Update Window Size and NDN network parameters such as topology and cache size.

Chapter 5

PPCSA Cache Replacement Strategy

In this chapter, a cache replacement strategy named Popularity Proportional Cache Size Allocation (PPCSA)[1] strategy has been proposed, which is designed for streaming content delivery on NDN. Unlike other previous works in this area, which have focused on the popularity of individual content chunks, in this study, I not only consider the file-level popularity of a streaming file which is composed of a set of video segments but also take the chunk-level probability of streaming chunk within a streaming file into consideration. I evaluated the strategy through simulations, by using a simple 4-level tree topology and a real network based hybrid topology. The results validate that on both scenarios, the proposed approach can increase the average cache hit ratio and shorten the average retrieval distance between clients and object contents. In addition, the energy efficiency of PPCSA strategy is analyzed, and I found that compared to other three cache replacement strategies, LRU, LFU, and FIFO, PPCSA reduces the transmission energy consumption, and improves the energy efficiency.

[1]This work has been published in the proceedings of European Conference on Networks and Communications IEEE, 2015, as the No. 3 paper listed in the list of publications.

5.1 Introduction

With the features of in-network caching, NDN is capable of facilitating multicast of contents and shortening the content delivery distance between content copies and requesters, by which the unnecessary network traffic can be reduced. Thus an appropriate cache management scheme could improve the network performance greatly.

By suggesting the new architecture of the future network, NDN provides various challenges for researchers to study. And there are many pieces of research existing in the study of cache strategy and video delivery.

A holistic model [76] has been developed to analyze the performance and overhead of cache management strategies in NDN. Based on this model, they proposed an optimal cache strategy named Coordinating In-network Caching to improve the overall NDN network performance and optimize the cache management cost. An in-network caching scheme ProbCache [37] has been introduced, which aims to reduce the cached redundancy on the content transmission path, and fairly distributes contents for multiple requesters on this path. Martin Draxler and Holger Karl proposed off-path based cache strategies [46] to avoid redundant caching of contents, so as to improve the utilization of cache space. These researches contribute in improving the cache efficiency of NDN in various aspects. However, the requested objects in their studies is the general contents. As for multiplexing a set of ordered contents, e.g. video segments, the cache management scheme needs to be reconsidered.

Cache replacement policy aims to carefully choose an appropriate content in CS to evict in order to make room for incoming contents. Although many efficient cache replacement policies such as LRU, LFU, FIFO have been widely adopted, the researches of seeking the more efficient policy of NDN is being carried out. Two cache replacement algorithms have been proposing to enhance the efficiency of H.264/SVC based video streaming delivered in NDN [74], while the scalability of these algorithms is under discussion. A Reuse time (RT) [85] based caching policy has been presented, which is a practical extension study of MIN algorithm[20]. RT caching policy exploits the periodicity of request stream pattern in the case of video streaming and exactly predicts the reuse time of a segment by the knowledge of play start time of each user to view this video. However, the overhead of predicting reuse times for every video segment has not been clearly assessed.

In this chapter, I propose a cache replacement strategy named PPCSA, for improving the cache performance of streaming content delivery over NDN. Based on the features of streaming content request model, I not only considered the file-level popularity of each streaming file which consists of a group of streaming chunks but also took the chunk-level probability of streaming chunks into account. PPCSA has been evaluated in the simulation on two topologies with variable cache sizes. And the results show that compared with LRU, LFU, and FIFO, the proposal enhances the network performance of NDN in several aspects. In addition, the result of energy efficiency analysis indicates that PPCSA is more energy efficient compared to the other three cache replacement strategies.

5.2 Popularity Proportional Cache Size Allocation Strategy

5.2.1 Assumptions

In NDN, contents are requested according to its individual names, and the name structure used in NDN is hierarchical name structure, e.g.

/Prefix/$Video_i$*/*$Content_j$. Generally, a streaming file in NDN is composed of a set of streaming segments (chunks) which are requested in sequence while streaming is being requested.

I, therefore, assume that the name of requested streaming chunks in NDN includes a streaming title name i and a sequence number j which is used for distinguishing different chunks of this streaming file. And the sequence number j is ordered based on streaming chunk sequence. Accordingly, I define that a streaming file f_i consists of a set of streaming chunks $\{C_{i,1}, C_{i,2}, \ldots, C_{i,j} \ldots\}$, which are ordered by its sequence number j. Besides, in this study, I consider that streaming file will be requested from the first chunks to the last one.

5.2.2 PPCSA Model

PPCSA is a cache replacement strategy which is concerned with both the file-level popularity of streaming file and chunk-level popularity of streaming chunks. Moreover, based on the natural linear time structure of streaming chunks, the popularity of each chunk within a streaming refers to the probability of this content being requested in the future. The PPCSA model can be divided into two components: Cache Size Allocation and Evictee Selection, and the two types of popularity mentioned above are adopted.

Cache Size Allocation Algorithm

I argue that according to the file-level popularity, the allocation of storage space for each streaming file in the CS needs to rapidly respond to the dynamic variation of request ratio. Thus, I measure the request ratio Req_i for each file and calculate the real-time CS occupancy rate Rcs_i for file i cached in the CS, in order to accurately adjust the allocated cache size for each file while cache replacement occurred. And the calculation of Req_i and Rcs_i takes place as follows:

$$Req_i = \frac{\sum_{j=1}^{n} Rec_{i,j}}{\sum_{i=1}^{m} \sum_{j=1} Rec_{i,j}} \tag{5.1}$$

$Rec_{i,j} \in \{0, 1, 2, ...\}$ represents the observed number of requests for content $C_{i,j}$ in a single time unit. The m, n in Equation 5.1, are considered as the amount of requested files and the number of requested chunks respectively. Equation 5.1, request rate Req_i implies the dynamic popularity of file f_i.

$$Rec_i = \frac{\sum_{j=1} Rcs_{i,j}}{K} \tag{5.2}$$

In Equation 5.2, $Rcs_{i,j} \in \{0, 1\}$ indicates whether chunk $C_{i,j}$ is cached in this CS, and K is the CS cache capacity of a node. Therefore, the real time cache occupancy ratio for file f_i in CS can be expressed by Equation 5.2. Namely, Rec_i represents the proportion of cache size occupied by file f_i.

The goal is to allocate cache proportional to content popularity, i.e., $Req_i = Rec_i$. $Req_i < Rec_i$ indicates that file f_i is occupying more cache spaces than it needs. On the contrary, $Req_i > Rec_i$, I can regard that more chunks of file f_i need to be cached to meet the request ratio, in order to enhance the utilization of CS.

Evictee Selection Algorithm

Users request streaming chunks in the order of sequence number j, then I deduce that the subsequent chunk have higher probability to be requested in the future. For instance, if an NDN router receives a request for content $C_{i,6}$, the subsequent chunks of the same file such as $C_{i,7}$, $C_{i,8}$ etc. would likely be requested later. Therefore, any of these subsequent chunk which cached in

CS, will have a higher opportunity to be requested than prior chunks. And PPCSA aims to keep these subsequent chunks in the CS for the future requests by selecting the chunk with the smallest sequence number j to be the evictee candidate.

PPCSA Cache Replacement Strategy

PPCSA cache replacement strategy is able to adjust the CS space occupied by each file due to the consideration of file-level popularity while processing cache replacement algorithm. In the meantime, PPCSA strategy evicts the content which has smallest probability to be requested in the future.

When an NDN router receives a data chunk $C_{i,j}$, firstly the router compares the monitored file-level popularity Req_i and CS storage space occupancy rate Rec_i. And then based on the evictee candidate selection model, I choose the suitable content to evict, as shown in Figure 5.1.

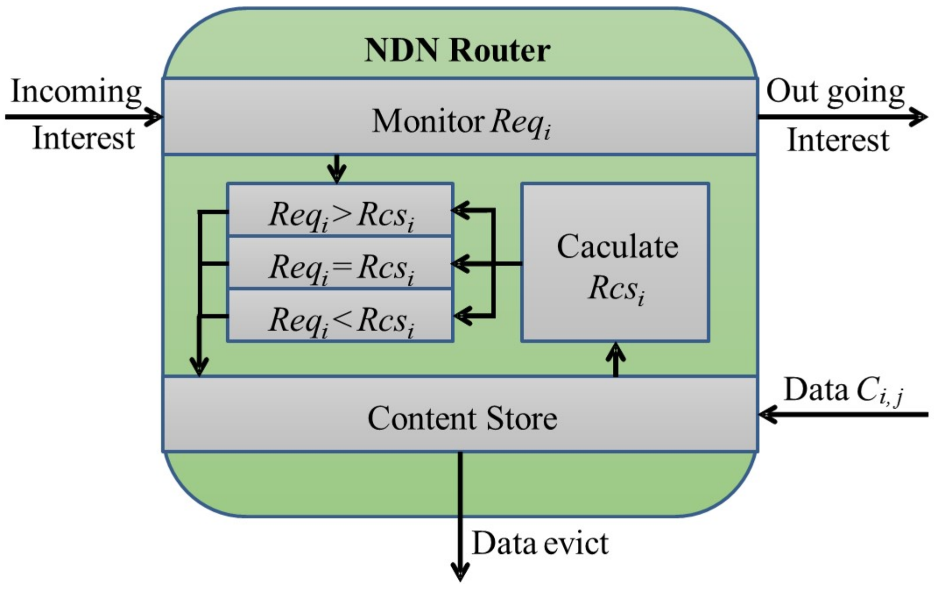


FIGURE 5.1: PPCSA cache replacement strategy.

If $Req_i < Rec_i$, file f_i is occupying CS rooms more than it needs. Thus some of cache space should be freed for other files to enhance the CS efficiency. Based on the evictee candidate selection method, I therefore choose two chunks $C_{i,p}$, $C_{i,q}$ of file f_i in the CS to evict, and the evict candidates $C_{i,p}$, $C_{i,q}$ have the smallest two sequence number ($p < q$), as shown in Figure 5.2.

If $Req_i = Rec_i$, I select only one chunk $C_{i,p}$ of file f_i in the CS, which has the minimum sequence number p, and evict it.

Otherwise, if $Req_i > Rec_i$, it refers that more contents of f_i need to be cached in this CS. Thus no chunk of f_i will be removed. Instead of that, I select the chunk $C_{k,p}$ to evict, which has the minimum sequence number p of file f_k, and this file has the smallest request rate Req_i in CS. Figure 5.2 illustrate the algorithm.

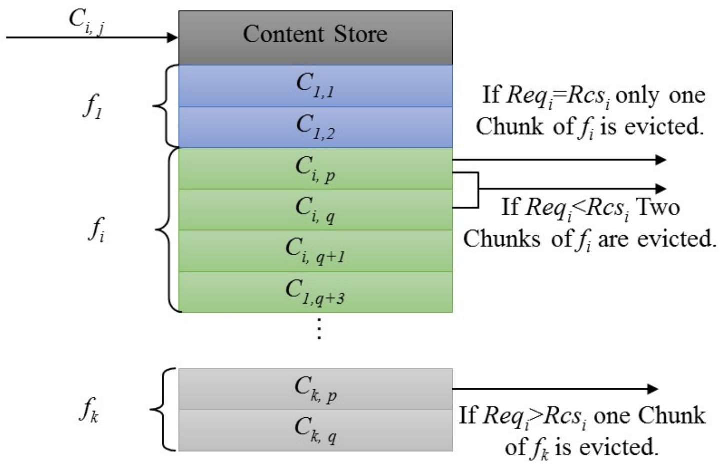

$C_{i,j}$
Content Store
f_1
$C_{1,1}$
$C_{1,2}$
If Req_i=Rcs_i only one Chunk of f_i is evicted.
f_i
$C_{i,p}$
$C_{i,q}$
$C_{i,q+1}$
$C_{1,q+3}$
If Req_i<Rcs_i Two Chunks of f_i are evicted.
f_k
$C_{k,p}$
$C_{k,q}$
If Req_i>Rcs_i one Chunk of f_k is evicted.

FIGURE 5.2: An example of PPCSA cache replacement algorithm.

5.3 Simulation and Results

I implemented the PPCSA strategies with ndnSIM [71] simulator. The proposed strategy is compared with 3 widely adopted chunk-level cache replacement strategies LRU, LFU, and FIFO on two topologies: a 4-level tree topology and a complicated hybrid topology. LCE, the default cache decision strategy, has been adopted in the simulation to work cooperatively with the cache replacement strategies.

5.3.1 Simulation Configuration

To evaluate the performance of each cache replacement strategy in a realistic network environment, video producers and clients were all connected to the edge of the topology. And I implemented 25 different producers in the simulation, and each of the producers provided a unique streaming file. In addition, every file is consisted of 800 video chunks. Accordingly the total chunk number is 20,000 items.

Furthermore, referring to file-level popularity, studies have shown that the popularity of streaming files follows Zipf-distribution [84, 86]. Thus 100 clients are implemented and the popularity of request for different files follows the Zipf-distribution and assumed that $\alpha = 1.2$. These clients started to request files with a given intervals which were assumed to follows the exponential-distribution. Besides, the consumers requested chunks in the order of the sequence number j from the beginning of this file to the end. Each simulation was stopped only if all 20,000 contents had been successfully received by their corresponding clients. And the simulation was performed independently across a range of CS sizes from 100 items to 1000 items for each NDN node.

I took two topologies to arrange the simulation. The first one is a 4-level tree in order to test the performance of the PPCSA strategy on a simple and small-scale network. To address the scalability and the performance on complex conditions of PPCSA strategy, another group of simulations were carried out arranged on a hybrid topology which composed of a core network and an access network, as shown in Figure 5.3. I chose GEANT [83] as the core network of the hybrid topology, and for the access networks I used 3-levels complete binary tree.

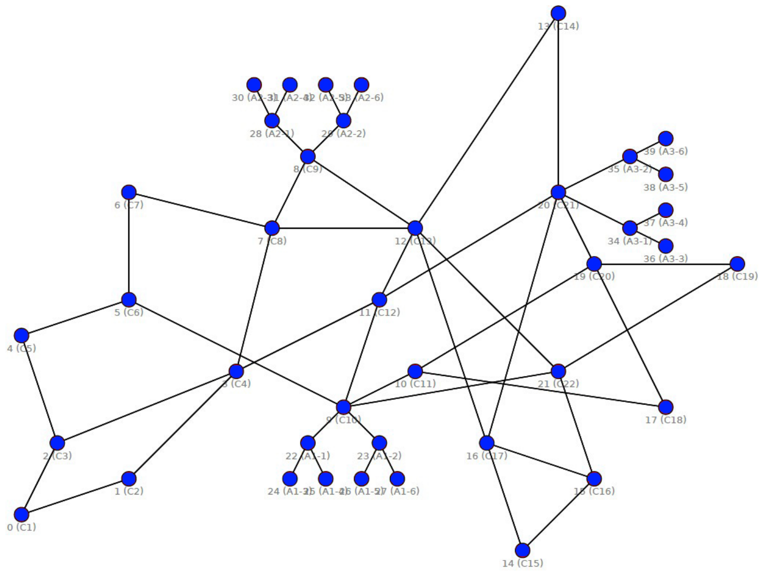


FIGURE 5.3: Hybrid topology.

5.3.2 Evaluation Parameters

4 values were used to evaluate the performance of each cache replacement strategies.

1. Total retrieval time illustrates the total processing time for all the 100 consumers to receive their requested videos contents.
2. Average retrieval distance indicates the average transmission round trip distance (number of hops) for each video chunk request.

3. Average cache hit ratio is pervasively used, as the primary performance metric to measure the caching performance.

4. Network traffic is the sum of total data size which has been forwarded by each node during the entire simulation.

5.3.3 Simulation Results

Results of cache replacement strategies arranged on tree topology are presented in Figure 5.4, 5.5, 5.6 and 5.7. PPCSA reduced the average retrieval distance and raised the average cache hit ratio as shown in Figure 5.5 and 5.6. It denotes that the PPCSA enhances the utilization of cache space so that requesters can get the streaming content from the closer routers, in other words, PPCSA lightens the pressure imposed on the original video sources. Furthermore, Figure 5.7 provides the evidence that PPCSA strategy is more efficient than the other three cache replacement strategies, due to unnecessary network traffic reduction.

Therefore, I can conclude that PPCSA strategy is able to significantly reduce the total retrieval time compared with LRU, LFU, and FIFO as shown in Figure 5.4. When CS size is more limited, the improvement is more obvious. I believe that there are two possible reasons for the finish time reduction. On one hand, PPCSA reduced the average retrieval distance between requesters and target content chunks so that the average transmission time of each file chunk is reduced. On the other hand, by reducing the unnecessary network traffic, the network congestion and packet loss occurred less than other algorithms. Then requesters do not have to frequently resend the request, and thus the total retrieval time can be reduced obviously.

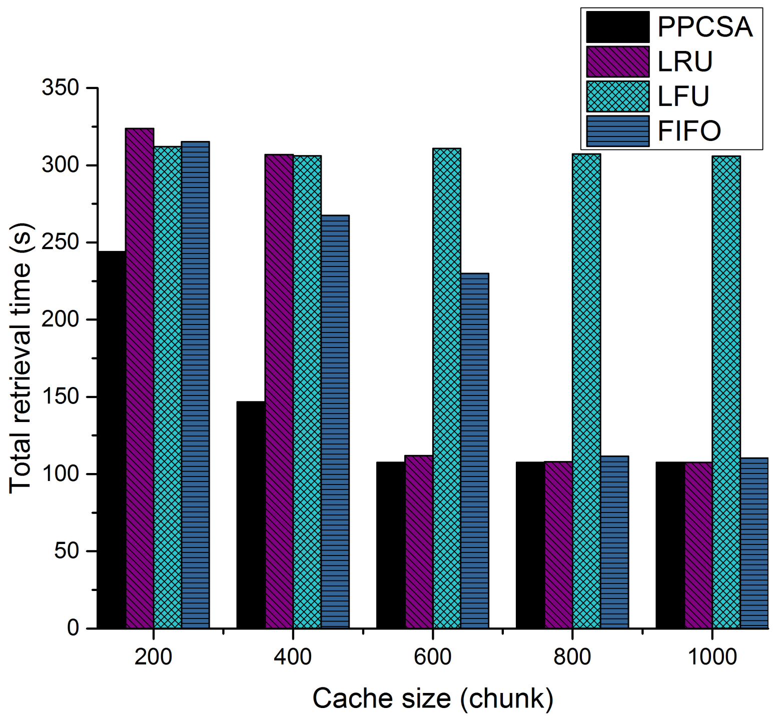


FIGURE 5.4: Total retrieval time on tree topology.

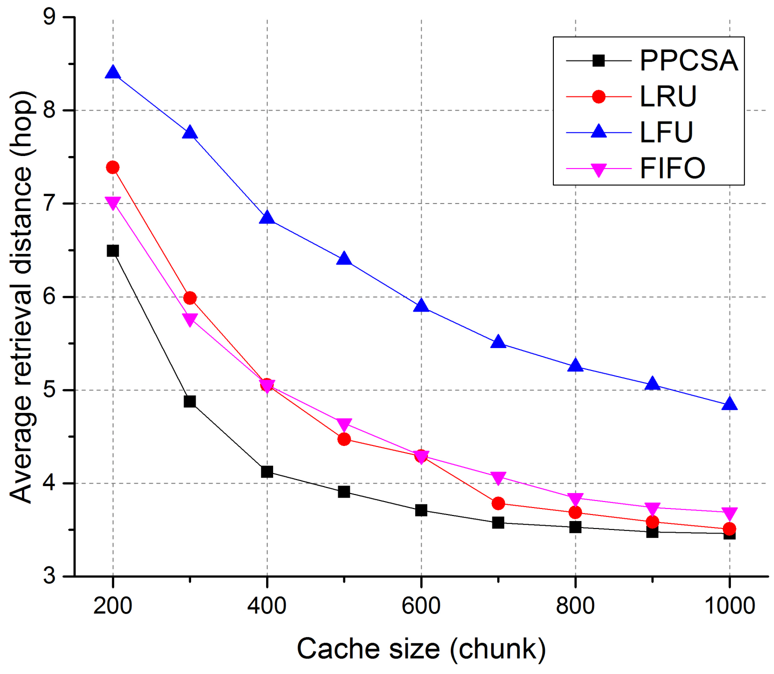


FIGURE 5.5: Average retrieval distance on tree topology.

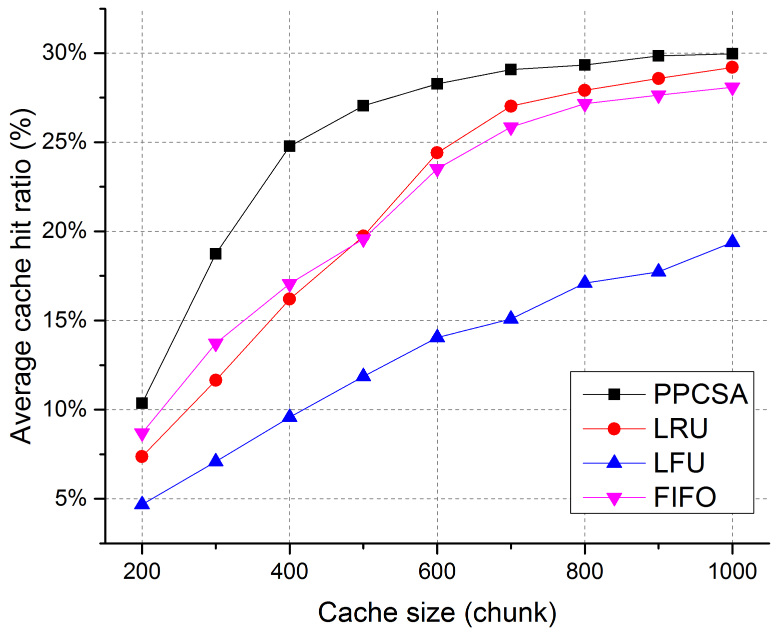


FIGURE 5.6: Average cache hit ratio on tree topology.

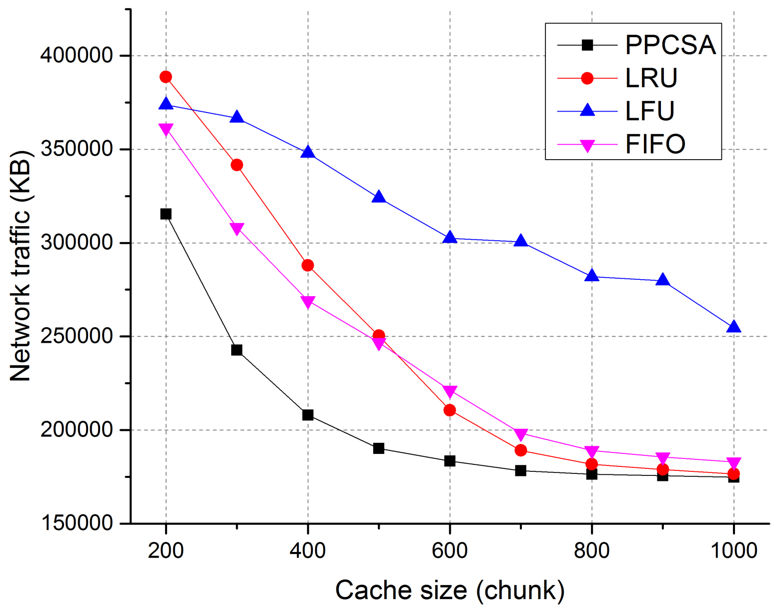


FIGURE 5.7: Network traffic on tree topology.

When PPCSA strategy was implemented onto a complex scenario, the improvement of network performances was even more, as shown in Figure 5.8, 5.9, 5.10 and 5.11. The improvement of total retrieval time and network traffic as shown in Figure 5.8 and 5.11 is more salient than the performance on the simple topology. PPCSA preforms independently on each router without further information of other nodes. The file popularity and cache occupancy ratio for each file are based on the real-time observation by each independent router, and thus the complexity of network topology cannot affect the performance of PPCSA. Therefore, based on the results, I can denote that the PPCSA strategy shows great scalability and also can be deployed to complex network scenarios.

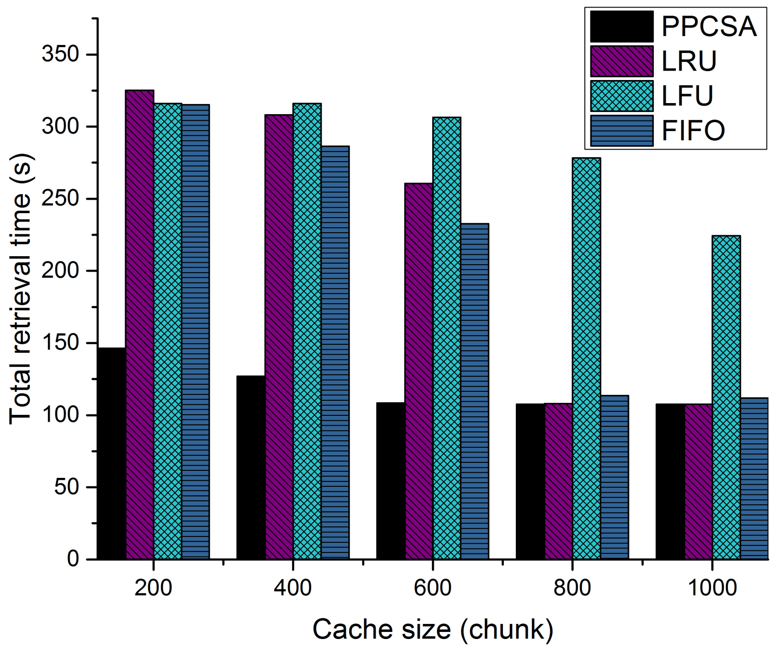


FIGURE 5.8: Total retrieval time on hybrid topology.

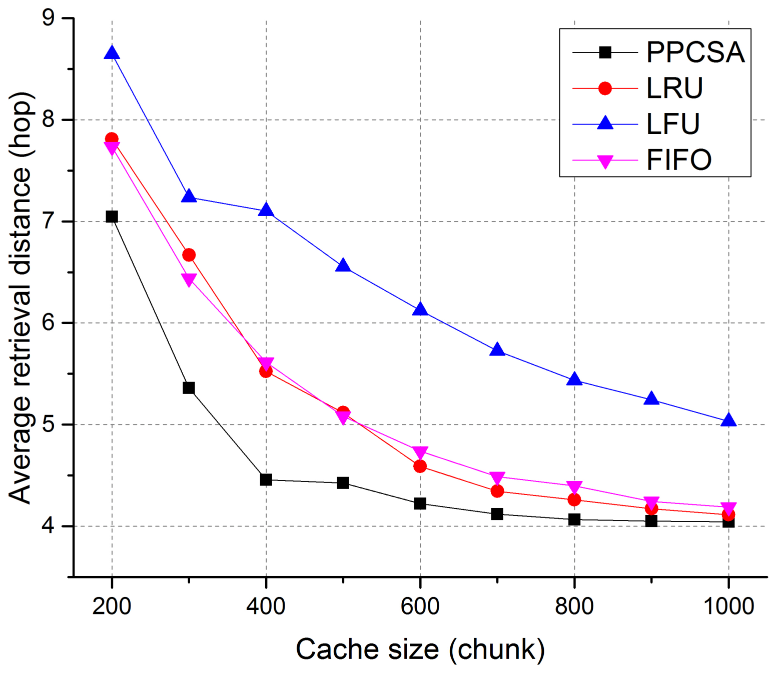


FIGURE 5.9: Average retrieval distance on hybrid topology.

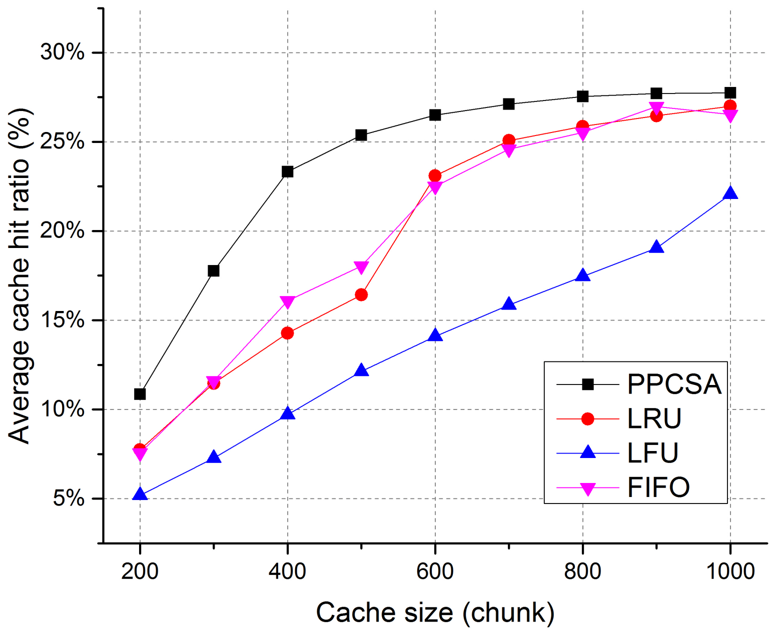


FIGURE 5.10: Average cache hit ratio on hybrid topology.

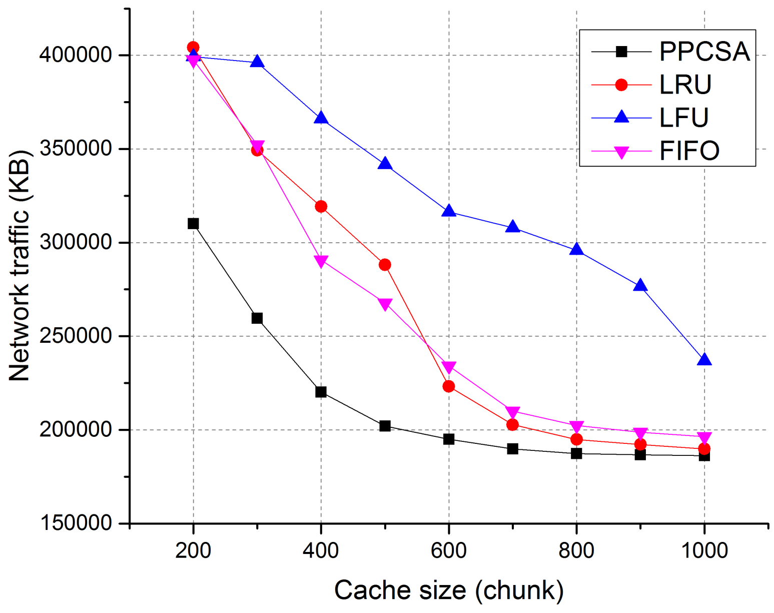


FIGURE 5.11: Network traffic on hybrid topology.

5.3.4 Energy Efficiency Analysis

It can be observed that many studies argued the power consumption of NDN [87, 88]. Researchers are currently making their efforts to reduce the energy consumption of NDN in different aspects. In the study, according to the simulation results, a power consumption model has been proposed, as shown below.

$$E_{NDN} = E_t + E_s + E_{idle} \tag{5.3}$$

$$E_t = P_t \times Network\ traffic \tag{5.4}$$

$$E_s = P_s \times Total\ cache\ size \tag{5.5}$$

$$E_{idle} = P_{idle} \times \textit{Total retrieval time} \tag{5.6}$$

The total power consumption of NDN expressed as E_{NDN} has three components, as shown in Equation 5.3: E_t indicates the transmission power consumption by NDN, and P_t (J/GB) represents the energy per size unit to transport data across the network. E_s refers to the storage energy cost while P_s (J/GB) is the cache energy consumed per GB. And E_{idle} refers to the inherent power consumed by the network during the whole simulation, and P_{idle} (J/s) represents the idle network power consumption per second.

According to the results, by reducing the total retrieval time, PPCSA is capable of reducing the E_{idle}. On the other hand, transmission power E_t can be cut down according to the reduction of network traffic. And the reduction percentage is shown in Figure 5.12. PPCSA reduces the transmission energy consumption over 15% on average compared to LRU and FIFO, and nearly 30% to LFU. And for E_{idle}, the reductions reach 34% compare to LRU and FIFO, and 60% from LFU.

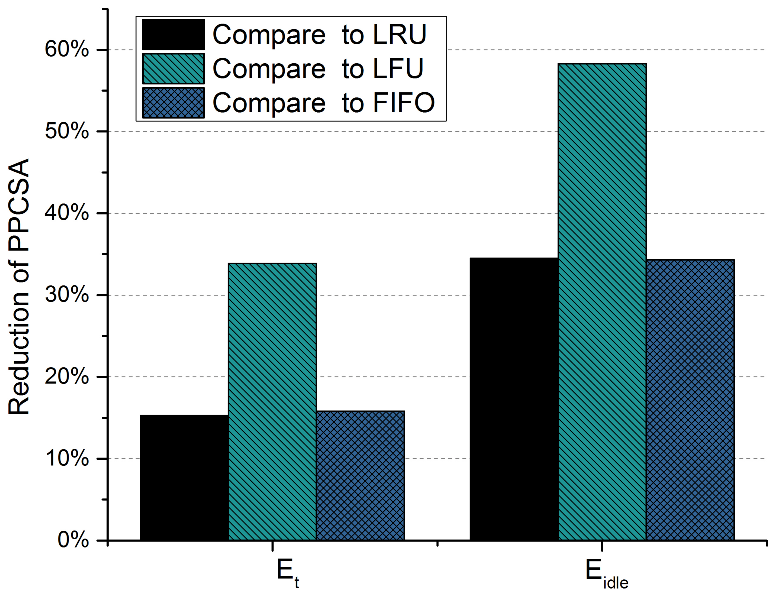


FIGURE 5.12: Power reduction percentage of PPCSA compare to LRU, LFU and FIFO.

5.4 Summery

By considering the characteristics of streaming content delivery, a cache replacement strategy PPCSA is proposed which is an effort to enhance the cache performance of NDN. In this chapter, I introduced the PPCSA model and strategy with both file- and content-level popularity considered. In the simulation, I implemented PPCSA strategy in ndnSIM simulator as well as LRU, LFU, and FIFO for comparison. And the results show that on both tree and hybrid topologies, PPCSA increased the cache hit ratio, shortened the total retrieval time, and average retrieval distance between requesters and contents, and showed great scalability and adaptability for complex network scenarios. In addition, the energy efficiency of above-mentioned cache replacement strategies are analyzed and the result shows that PPCSA is more energy efficient than other three cache replacement strategies.

Chapter 6

TLP-TTH Cache Replacement Sstrategy

I introduce a novel cache replacement strategy to improve the entire network performance of video delivery over NDN. In the case of the NDN structure, I argue that: 1) for video multiplexing scenario, general cache strategies that ignore the intrinsic linear time characteristic of video requests are unable to make better use of the cache resources, and 2) it is inadequate to simply extend the existing research conclusions of file-oriented popularity to chunk-by-chunk popularity, which are widely used in NDN.

Unlike previous works in this field, the proposed strategy in this chapter, named TLP-TTH[1] cache replacement strategy, is designed on the basis of the following principles. Firstly, the proposed cache replacement strategy is customized for video delivery by carefully considering the essential auto-correlated request feature of video chunks within a video file. Furthermore, the popularity in video delivery is subdivided into two levels, namely chunk-level access probability and file-level popularity, in order to efficiently utilize cache resources. I evaluated the proposed strategy in both a hierarchical topology and a real network based hybrid topology, and took viewers departure into consideration as well. The results validate that for video

[1]This work has been published in IEICE Transactions on Communications 99.12 (2016), as the 1st paper listed in the list of publications.

delivery over NDN, TLP-TTH strategy improves the network performance from several aspects. In particular, I observed that the proposed strategy not only increases the cache hit ratio at the edge of the network but the cache utilization at the intermediate routers is also improved markedly. Further, with respect to the video popularity variation scenario, the cache hit ratio of TLP-TTH strategy responds sensitively to maintain efficient cache utilization.

6.1 Introduction

The in-network caching mechanism allows intermediate network devices to temporarily store the passing content so that subsequent requests for these contents can be responded directly without visiting the original content source [37]. In-network caching mechanism enables every router to respond cached contents to the requests, which results in multiple content providers in the network. As a result, in-network caching facilitates the multicast of contents and shortens the content delivery distance between the requesters and the target contents. Thus, the overall unnecessary network traffic among intermediate routers can be reduced. As such, the management scheme of in-network caching will significantly affect the network performance.

In-network caching in NDN rises new challenges. Studies of traditional in-network caching such as web, P2P, etc. are file-oriented, and the unit of cache placement and replacement is a file. However, within the NDN infrastructure, the unit of caching is a fragment of a file called chunk. And this variation of operation unit brings new challenges [49]. On one hand, the popularity is defined for different granule size. The well-accepted researches of popularity are file-oriented, e.g. the request popularity follows the Zipf-distribution of web objects [49]. However, it is inadequate to simply extend the existing research conclusions of file-oriented popularity to chunk-by-chunk popularity, which are widely used in NDN. The reason is that users in NDN may get the whole file from different routers, and different chunks of a single file will have different access frequencies at an NDN router. On the other hand, *Independent Reference Assumption* which is typically used in traditional file-oriented caching is no longer appropriate to NDN [49]. Independent reference assumption refers that the request of a particular object is emerged independently and will not affect other objects. For video delivery over NDN, the request for different chunks within the same video are well ordered according to the playback sequence, rather than be generated independently. Therefore, in the case of chunk-by-chunk video content delivery, the feasibility of independent reference assumption needs to be reconsidered.

The in-network caching mechanism in NDN introduces distinctive challenges, and many researchers have devoted themselves to their studies from different criteria.

The new challenges have been outlined when the in-network caching shift from traditional scenario to NDN case. New features of NDN caching such as transparency, ubiquity, and fine granularity lead to the reconsideration of cache schemes [89–91]. And [92] introduces and classifies the studies of web cache strategies. Although these strategies may not be targeting NDN, they are excellent references for designing NDN cache policies.

One of the core concerns of NDN in-network caching is its cache decision policy, which focuses on choosing the appropriate caching location for particular contents in the network. The decided cache location could be along the transmission path [36, 46, 93] or might not be limited to the content delivery path [50, 90]. Other studies such as [35, 94] focus on reducing network congestion and improving network performance on the user-centric performance metrics.

A cache replacement strategy is designed for carefully selecting and evicting the contents of the a cache in order to make storage space for the new incoming data. For web caching scenario, cache replacement issues are well studied [95–98]. Several papers such as [85] proposed the cache replacement policies specific for video delivery. In NDN architecture, [99] developed a dynamic classification method to reduce the time complexity of cache inquiry and proposes a fast convergence caching replacement algorithm named FCDC. Nevertheless, the abovementioned study concentrates on distributing general independent contents. For a set of organized contents, e.g. video chunks, the cache scheme has to be reconsidered. In [74], a cache replacement algorithm for layered video content has been proposed. However, the experiment was conducted on a simple and liner network with 5 nodes, thus the scalability of this proposal is unclear.

The independent reference model is widely adopted for evaluating the performance of caching policies [100, 101], according to its simplicity and effectiveness. However, only taking independent reference assumption into consideration is insufficient for reflecting request temporal locality and spatial locality [102], and may lead to evaluation errors [103]. [103] also presents a traffic model to account for the temporal locality observed in real YouTube traffic, and [104] introduces the geographic spatial locality of YouTube videos. Nevertheless, these traffic observations are based on TCP/IP traffic and the target object is the video file. To better evaluate the caching performance for NDN, chunk-level temporal locality and spatial locality need to be considered.

In this study, I address the problem of cache replacement, which focuses on video delivery over NDN. The research goal is to enhance cache utilization and reduce unnecessary network traffic by proposing TLP-TTH cache replacement strategy. The popularity used in this study is split into two levels, chunk-level access probability and file-level popularity. And the two algorithms of TLP-TTH strategy adopt each level of the popularity. In addition, instead of using independent reference assumption, the cache replacement strategy for video delivery has been redesigned by considering the auto-correlation of the request sequence within a video file and the effect of subsequent requests. Besides, the cache resources allocated to each video file should be rapidly adjusted to adapt the dynamically fluctuant local file-level popularity. And I also attempt to avoid additional management overhead, such as messages or control packets, which will increase the network burden.

6.2 Video Delivery Scenario over NDN

NDN communications rely on two types of packet, interest and data. The interest packet is designed for requesting a target content and the data packet is for delivering the requested content. The content is requested according to its name, and the name structure used in NDN is hierarchical, (e.g. */Prefix/Video_name/Chunk_name*). In general, a video file is structured in video chunks, which are transferred in sequence while the video is being played. And each video chunk has a unique name, which would allow the chunk to be requested, forwarded, and cached independently.

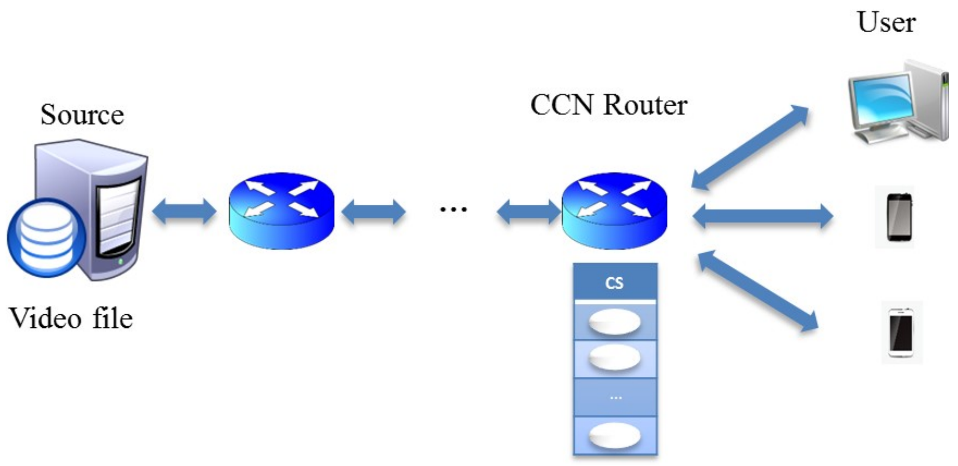


FIGURE 6.1: Scenrio of video delivery over NDN.

The scenario of video delivery over NDN is illustrated in Figure 6.1. When an NDN router receives an interest packet, the Content Store (CS) will be traversed by the content name carried by the interest packet in order to find whether the target chunk is cached there. If this chunk exists in the CS, it will be packaged into a data packet to be sent to the client directly. This is called a cache hit. Otherwise, the interest packet will be forwarded to upstream routers and lead to a cache miss at this router. The video chunk in a data packet will be cached in the CS while NDN routers are forwarding the data packet, and if the cache space is exhausted, a selected chunk will be evicted to make room for the incoming data.

A cache replacement strategy is designed not only for finding a evictee that makes room for the new coming chunk, but also for rationally managing the CS space of NDN routers, in order to enhance the utilization efficiency of the cache space globally. As a result, a well designed cache replacement strategy can improve the network performance markedly, in terms of the average cache hit ratio, the average chunk delivery distance, etc.

6.3 TLP-TTH Cache Replacement Strategy

Instead of only taking file-oriented popularity into consideration, I introduce two levels of popularity, namely chunk-level access probability and file-level popularity. In this study, The chunk-level access probability indicates the local access probability among video chunks within a particular video file, and the file-level popularity expresses the realtime local access probability among different video files. The TLP-TTH cache replacement strategy is composed of *Time-To-Hold* evictee selection algorithm and *Cache Size Allocation* algorithm, which take the two levels of popularity mentioned above into consideration. In addition, although the file level requests emerge independently, the requests for chunks within a video file are well ordered in sequence. Therefore, I believe that instead of independent reference assumption, for chunk-by-chunk communication in NDN, auto-correlation of the request sequence within the video needs to be considered.

6.3.1 Definitions and Assumptions

I make the following definitions and assumptions:

1. The name of the video chunk in NDN is abstracted as two components, the video file name *i* and the sequence number *j* which can be used for distinguishing different chunks of this video file (e.g. *movie03/0009*).
2. The sequence number *j* is ordered based on the video playback sequence. (e.g. *the sequence number of the first chunk is 0001, and the second is 0002 and etc.*).
3. Accordingly, I define that a video file f_i consists of a set of video chunks $\{C_{i,1}, C_{i,2}, ..., C_{i,j}...\}$, which are ordered by the sequence number j, and each chunk $C_{i,j}$ is assumed to have the same chunk size.
4. Video chunks will be requested in sequence from the first chunk.

6.3.2 Time-Ho-Hold Evictee Selection Algorithm

Since I assume that consumers will request the video chunks in a row according to the sequence number, I deduce that the subsequent chunks will have a relatively high probability of being requested in the future. For instance, if a NDN router receives an interest packet for requesting *movie01/0002*, the subsequent chunks such as *movie01/0004* and *movie01/0007* are likely to be requested later. Therefore, if any of these subsequent chunks has already been cached in the CS, they will have a higher chance being requested than other chunks, i.e. they will have high chunk-level access probability. Further, the research aim is to "hold" these chunks in the CS for a short time span in order to ensure that these chunks will not be easily replaced by the cache replacement strategy before they are requested.

$tth_{i,j}$ is the estimated time point until which the already cached chunks are expected to be stored in the CS. An early $tth_{i,j}$ value implies that the corresponding chunk has low probability of being requested in the the future. In contrast, a late $tth_{i,j}$ value indicates high chunk-level access probability, and the corresponding chunk will probably keep being requested for a relatively long period of time. Therefore, when the cache replacement procedure is executed, the evictee should have the minimum $tth_{i,j}$ as compared to the other chunk within a particular video file.

I record the $tth_{i,j}$ value for every chunk in the CS, and update it promptly. When the chunk is initially cached in the CS, the $tth_{i,j}$ value of this chunk equals to the cached time t_{Cached}, as shown in Equation 6.1.

$$tth_{i,j} = t_{Cached} \tag{6.1}$$

Further, the $tth_{i,j}$ value should be updated when the NDN router receives other interest packets of this video. And for the $tth_{i,j}$ updating, I take the interrelationship between video chunks into consideration. According to the liner requesting feature of video delivery, the received interest implies that some cached subsequent chunks will likely be requested in the future. Thus, I stipulate that the $tth_{i,j}$ value of the cached chunks should be updated under the following constraints:

1. $tth_{i,j}$ updating only occurs when an NDN router receives an interest packet for video file f_i.
2. The sequence numbers of the cached chunks are larger than the requested chunk of the received interest packet.

The $tth_{i,j}$ value is updated on the basis of the sequence number gap between the just received interest packet and the cached chunks, as shown in Equation 6.2.

$$tth_{i,j} = \max\{tth_{i,j}, t_{Interest} + (j - j_{Interest}) \cdot \Delta t\} \tag{6.2}$$

Here, $t_{Interest}$ and $j_{Interest}$ denotes the received time point and the sequence number of chunk $C_{i,j_{Interest}}$, which is requested by the just received interest packet. j refers to the sequence number of cached chunk $C_{i,j}$. The playback duration of each video chunk is expressed by Δt. Δt is proportional to the chunk size and inverse proportional to the bit rate of this video. I assumed that in this study the video files have the same bit rate and chunks have a uniform size for simplicity as the initial study. Since Δt is defined for each video file, information to derive Δt may be included in the file name as an alternative.

When the NDN router receives an interest packet of video file f_i at the time $t_{Interest}$, the $tth_{i,j}$ value of the cached chunks within this video will be checked for an update. Consequently, the chunks that have higher chunk-level access probability will have a larger $tth_{i,j}$ and will be updated more frequently than the others which have lower probability of being requested in the future. The update procedure is presented in Algorithm 1, where i_k and $i_{Interest}$ represent the video file name of the cached chunk and the received interest, and j_k denotes the sequence number for the cached chunk. Further, K indicates the cache capacity (here I use the number of chunks to address the cache capacity) and $k \in [0, K)$ identifies the cached chunk in this CS.

For a particular video, the chunk with the minimum $tth_{i,j}$ value refers that this chunk has the lowest probability of being requested in the future. Therefore, when I need to replace chunk in the CS, one or two chunk(s) with the minimum $tth_{i,j}$ value have to be evicted.

Algorithm 6.1 $tth_{i,j}$ Updating

```
Require: $C_{i_{Interest},j_{Interest}}$, $t_{Interest}$, $K$
  Receive an interest packet $C_{i_{Interest},j_{Interest}}$
  for $k$ =0 to $K$ do
    if $i_k$ = $i_{Interest}$ then
      if $j_k > j_{Interest}$ then
        $tth_{i,j_k} = \max\{tth_{i,j_k}, t_{Interest} + (j_k - j_{Interest}) \cdot \Delta t\}$
      end if
    end if
  end for
```

6.3.3 Cache Size Allocation Algorithm

I argue that at each NDN router, the cache space allocated to each video file should be adjusted to the variation of the local file-level popularity, i.e. local real time file-level access probability. To achieve this goal, I monitor the request ratio Req_i for each video file f_i at each router to obtain the local real time access probability, and I calculate the current CS occupancy rate Rcs_i for every already cached video f_i. Rcs_i represents the percentage of storage space already occupied by video f_i in this particular CS. Further, Req_i and Rcs_i are calculated as follows:

$$Req_i = \frac{Rec_i}{Rec} \tag{6.3}$$

$$Rcs_i = \frac{\sum_{j=1}^{n} Rcs_{i,j}}{K} \tag{6.4}$$

Rec_i refers to the number of interest packets for video file f_i, which is measured in a unit of time, and Rec indicates the total number of request received by this node in the same time unit. Thus, according to Equation 6.3, request rate Req_i expresses the real time local access probability of video file f_i. Rcs_i indicates the current cache occupancy ratio for video file f_i in CS. In Equation 6.4, $Rcs_{i,j} \in \{0, 1\}$ addresses whether chunk $C_{i,j}$ is cached in this CS or not, and K denotes the cache capacity of this CS (number of chunks). By determining Req_i and Rcs_i of every cached video file, I can allocate the local cache resources to video files fairly and accurately. Besides, the calculation time unit will affect the value of Req_i to truly reflect the variation of file-level popularity, which will be changed by the arrival of new viewers. Therefore, I stipulate that, the time unit for these calculations is equal to the average time interval for new viewer arrival.

The TLP-TTH cache size allocation algorithm modifies the CS space occupied by each video file while cache replacement be executed. When an NDN router receives a data packet of chunk $C_{i,j}$, I compare the monitored Req_i and calculated Rcs_i of video file f_i. And according to the comparison result, I then face one of the three conditions presented below as shown in Figure 6.2.

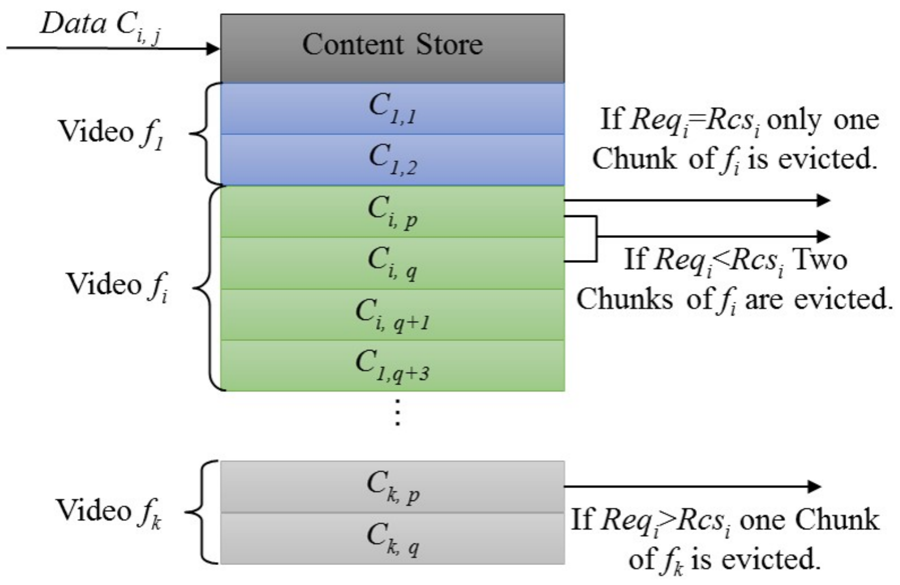

Data $C_{i,j}$
Content Store
Video f_1
$C_{1,1}$
$C_{1,2}$
If Req_i=Rcs_i only one Chunk of f_i is evicted.
Video f_i
$C_{i,p}$
$C_{i,q}$
$C_{i,q+1}$
$C_{1,q+3}$
If Req_i<Rcs_i Two Chunks of f_i are evicted.
Video f_k
$C_{k,p}$
$C_{k,q}$
If Req_i>Rcs_i one Chunk of f_k is evicted.

FIGURE 6.2: Example of TLP-TTH cache allocation algorithm.

$Req_i < Rcs_i$ shows that video f_i have occupied more CS space than it requests. Thus, some of the cache space needs to be freed for other video files in order to enhance the CS utilization efficiency. I decide to reduce the storage space for f_i, and according to the TLP-TTH evictee selection algorithm, I choose two chunks $C_{i,p}$ and $C_{i,q}$ to evict where the $tth_{i,p}$ and $tth_{i,q}$ are minimum of video f_i, and $C_{i,j}$ is stored at one of the vacated cache slots.

$Req_i = Rcs_i$ refers that the current cache size allocated to video f_i just fits the local file-level popularity. Then, the space taken by f_i will not be changed. Thus, only one chunk $C_{i,p}$ of video file f_i with the minimum $tth_{i,p}$ will be evicted during cache replacement.

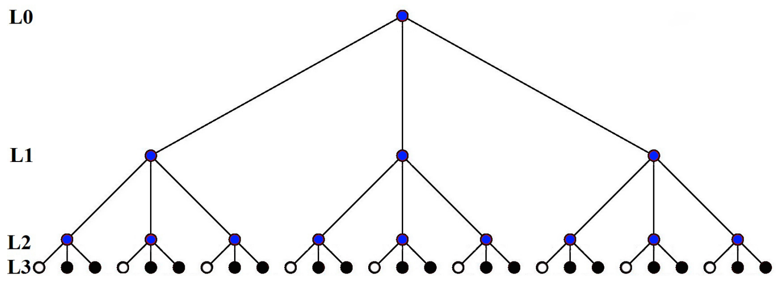
Lastly, $Req_i > Rcs_i$ indicate that more chunks of video f_i need to be cached in this CS. Thus, no chunk of video f_i will be replaced. Instead, according to the TLP-TTH evictee selection algorithm, I choose one evictee $C_{k,p}$ that belongs to the video file f_k, which has the smallest request ratio Req_k in the CS compared with other video files.

Because of the high frequency of cache replacement being processed in each router, the cache size allocation will be adjusted rapidly without an additional communication overhead.

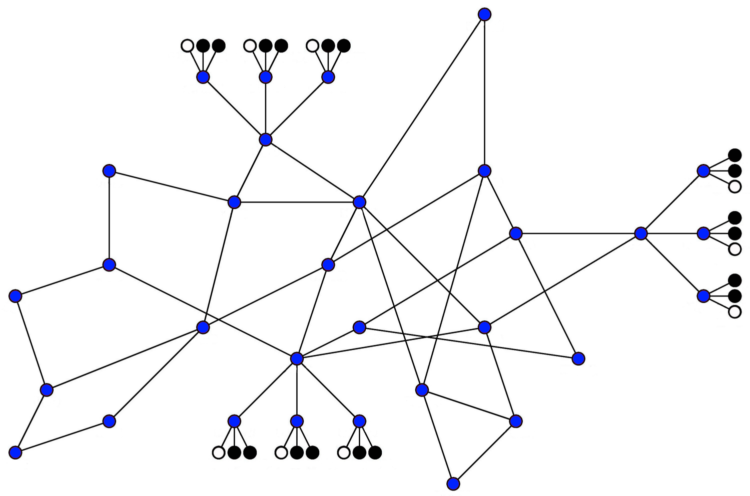
6.4 Simulation and Results

I compared the proposed scheme with other cache replacement strategies: LRU, LFU and my previous work PPCSA. I believe that any cache decision strategy such as LCD [104], Prob [72], ProbCache [37], etc. can work cooperatively with TLP-TTH due to the independence between cache decision strategy and replacement strategy. To clearly address the performance of TLP-TTH, I adopted the default cache decision strategy in NDN named Leave Copy Everywhere (LCE) to cooperatively work with the proposed cache replacement strategy.

TLP-TTH is evaluated with other cache replacement strategies by using two topologies: a 4-level complete ternary tree topology in which level L0 refers to the root level and level L3 represents the leaf level, as shown in Figure 6.3(a); and a hybrid topology consisting of a core network and access networks, as shown in Figure 6.3(b). For the core network, GEANT [83] was selected, and for the access networks, I used 3-level complete ternary tree.



(a) Hierarchical topology.



(b) Hybrid topology.

FIGURE 6.3: Simulation topologies.

I customized the two-level popularity-oriented request model for the simulations. To evaluate the performance similar to the real network circumstance, video producers are connected to the edges of the access networks. I implemented 9 producers uniformly allocated at leaf nodes, each sub tree has 3 producers, as the white nodes shown in 6.3. I implemented 25 different video files randomly in these producers, and each video file contains 1,000 video chunks that could be requested as NDN content. Accordingly, the total chunk number was 25,000 items. With respect to file-level popularity, studies have shown that it follows the Zipf-distribution [84, 86]. Therefore, I implemented 100 video viewers on the rest leaf nodes, and the popularity of requests for different video files followed Zipf-distribution (α = 1.2) [105]. In addition, these viewers started to request videos with a given intervals which were assumed to follow Exponential-distribution, and the mean is 1s. Besides, as I assumed, the consumers requested video chunks in the order of video sequence number j from the beginning of this video. In scenario 1, 2 and 4, viewers will not stop requesting till the end of the video, while in scenario 3 viewers may stop watching video before end. And the simulation was performed independently across a range of CS sizes from 200 items to 1000 items for each NDN node.

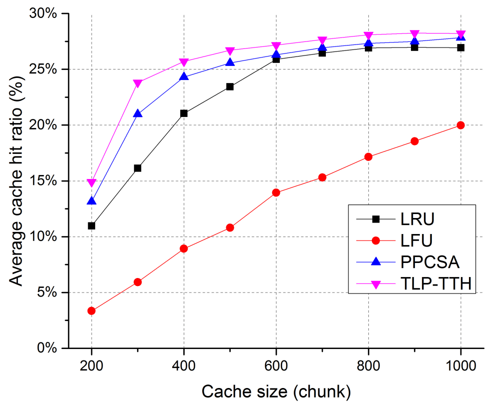
Four metrics were adopted to evaluate the performance of each strategy.

1. *Average cache hit ratio* was used as the primary performance metric to measure the cache utilization efficiency.
2. *Average retrieval distance* indicated the average transmission round-trip distance (the sum of the hops of one pair of interest and data packets transferred) for each video segment delivery.
3. *Network traffic* presented the amount of total data forwarded by every node during the entire simulation.
4. *Average retrieval time* denoted the average time period between interest packet sent and data packet successfully received.

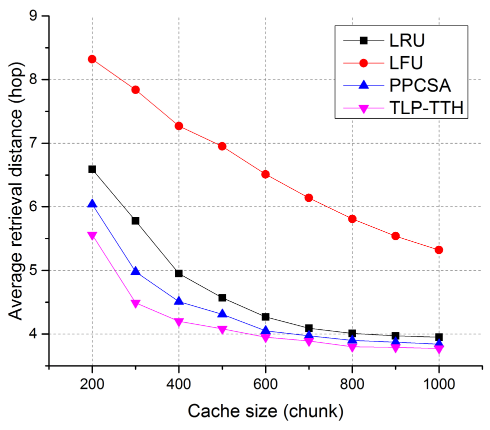
I evaluated the performance of TLP-TTH in the following four different scenarios.

6.4.1 Scenario 1: Hierarchical Topology with Multiple Producers

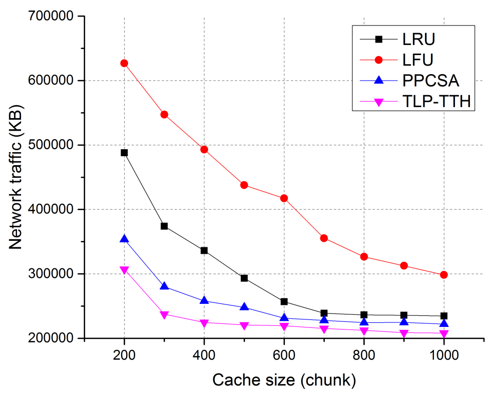
In this scenario, I tested the performance of the cache replacement strategies in a 4-level complete ternary tree topology.



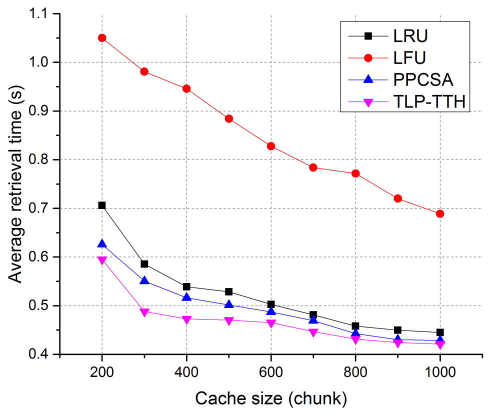
(a) Average cache hit ratio.



(b) Average retrieval distance.



(c) Network traffic.



(d) Average retrieval time.

FIGURE 6.4: Simulation results of hierarchical topology.

The simulation results presented in Figure 6.4(a) and 6.4(b) show significant differences among the LRU, LFU, PPCSA, and TLP-TTH strategies. In terms of average cache hit ratio, TLP-TTH outperforms the PPCSA strategy by 3%, while PPCSA shows 5% improvement over LRU. Therefore, the overall performance rise is about over 8% for TLP-TTH as compared with LRU and much more compared with LFU. Improvement in average retrieval distance is more impressive. TLP-TTH shows 24% decrease in average transfer distance compared with PPCSA, and nearly 30% decrease from LRU. This implies that compared with the reference strategies, TLP-TTH enhances the utilization of the cache space, and thus, viewers can acquire video chunks from the closer routers. In other words, TLP-TTH lightens the pressure of the original video sources.

TLP-TTH strategy reduces the total network traffic and the average retrieval time compared with LRU, LFU, and PPCSA, as proven by Figure 6.4(c) and 6.4(d). On one hand, TLP-TTH reduces the network traffic by 10% from PPCSA and 24% from LRU. On the other hand, for retrieval time TLP-TTH outperforms the PPCSA strategy with over 4%, and over 10% improvement form LRU on average. Because of the reduction of the average delivery distance between the viewers and the target content chunks, the unnecessary network traffic among intermediate routers has been reduced, as well as the average retrieval time. These reductions are further intensified by the high cache hit ratio of the TLP-TTH strategy. In conclusion, to deliver the same amount of video chunk, the TLP-TTH scheme consumes less network resource and takes less time with respect to LRU, LFU, and PPCSA, and when the cache size is limited (e.g. cache size $<$ 700items), the improvement is more evident.

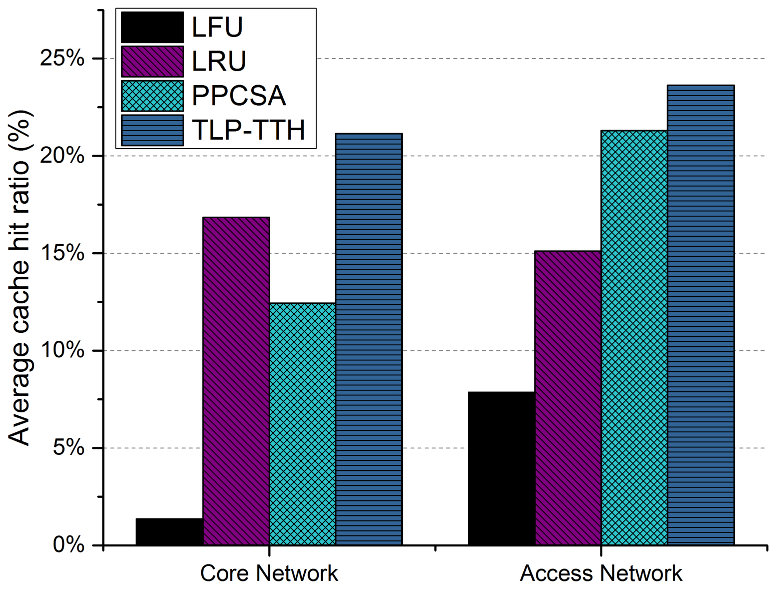


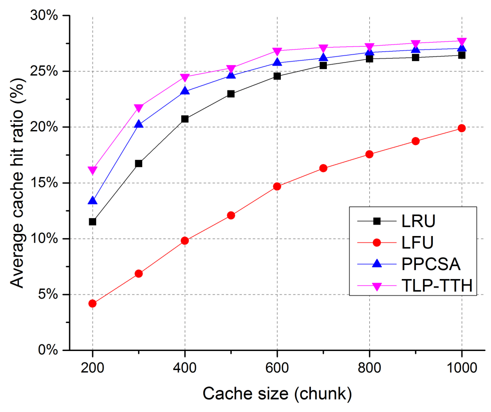
FIGURE 6.5: Average cache hit ratio in different domains of hierarchical topology.

In addition, I was curious about how TLP-TTH performed in different domains of a network. Therefore, I recalculated the cache hit ratio of scenario 1 (cache size=300 items) in different network regions. In the 4-level tree topology, the core network refers to levels L0 and L1, and the access network includes L2 and L3.

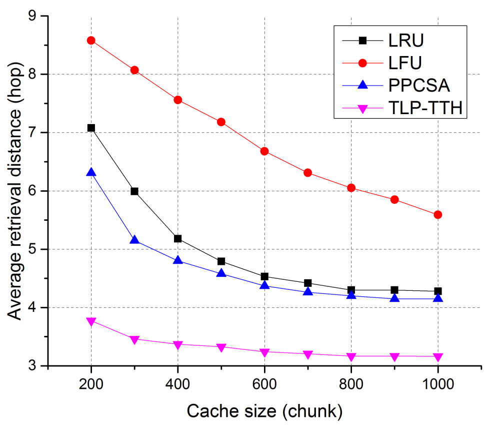
One interesting observation is that TLP-TTH increases the cache hit ratio at the edge of the network, as shown in Figure 6.5, as well as the cache hit ratio at the intermediate routers allocated at the core network. I believe that the improvement in the core network is mostly due to the evictee selection algorithm. I can see from Figure 6.5 that upon the adoption of the time to hold evictee selection algorithm, the cache hit ratio in the core network improved by 67% compared with PPCSA.

6.4.2 Scenario 2: Hybrid Topology with Multiple Producers

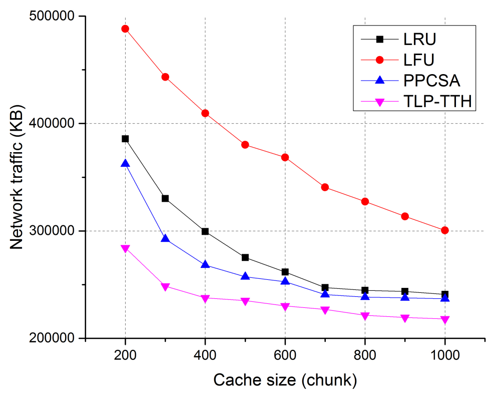
In this scenario, I aim to present the network performance of cache schemes on a complex hybrid network topology. I spread 9 source producers and 100 video viewers connected to the leaf nodes at each access network. When the TLP-TTH strategy is applied to this hybrid topology scenario, the improvement of the network performance is still marked, as shown in Figure 6.6.



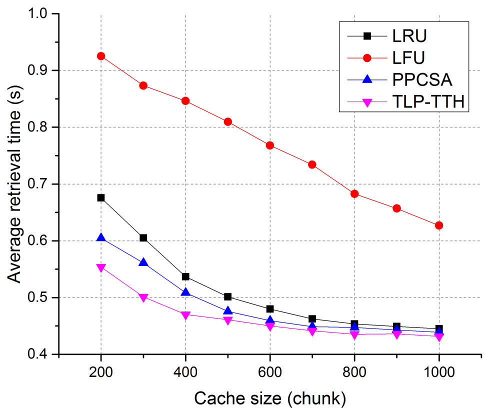
(a) Average cache hit ratio.



(b) Average retrieval distance.



(c) Network traffic.



(d) Average retrieval time.

FIGURE 6.6: Simulation results of hybrid topology.

Since the average cache hit ratio and the average retrieval distance are improved by TLP-TTH compared with other 3 strategies, the unnecessary network traffic and average content retrieval time are significantly reduced. TLP-TTH executes independently at each router without further information of other nodes or any additional management overhead. The file-level popularity and the cache occupancy ratio are based on real time observations and calculations of each independent router. Thus, the complexity of a network topology cannot affect the performance of TLP-TTH. On the basis of these results, I can conclude that the TLP-TTH strategy shows great scalability and can be deployed in complex network scenarios.

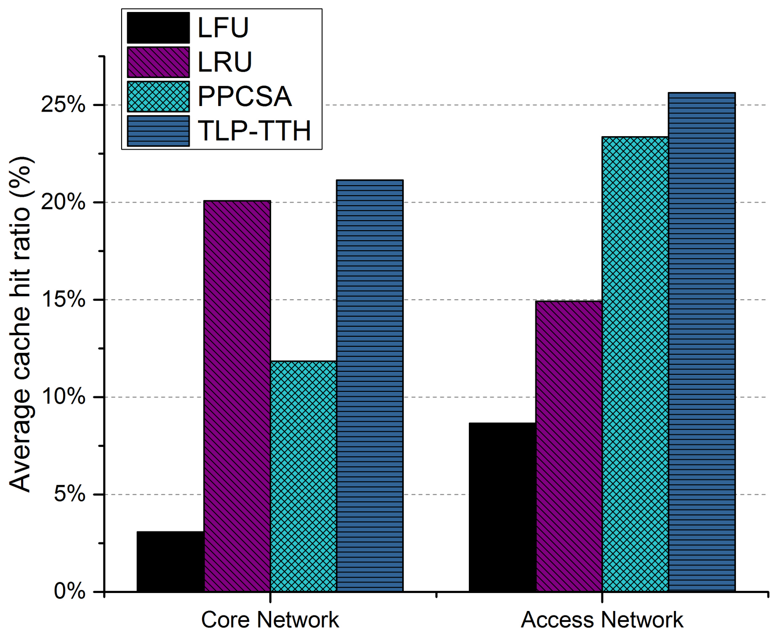


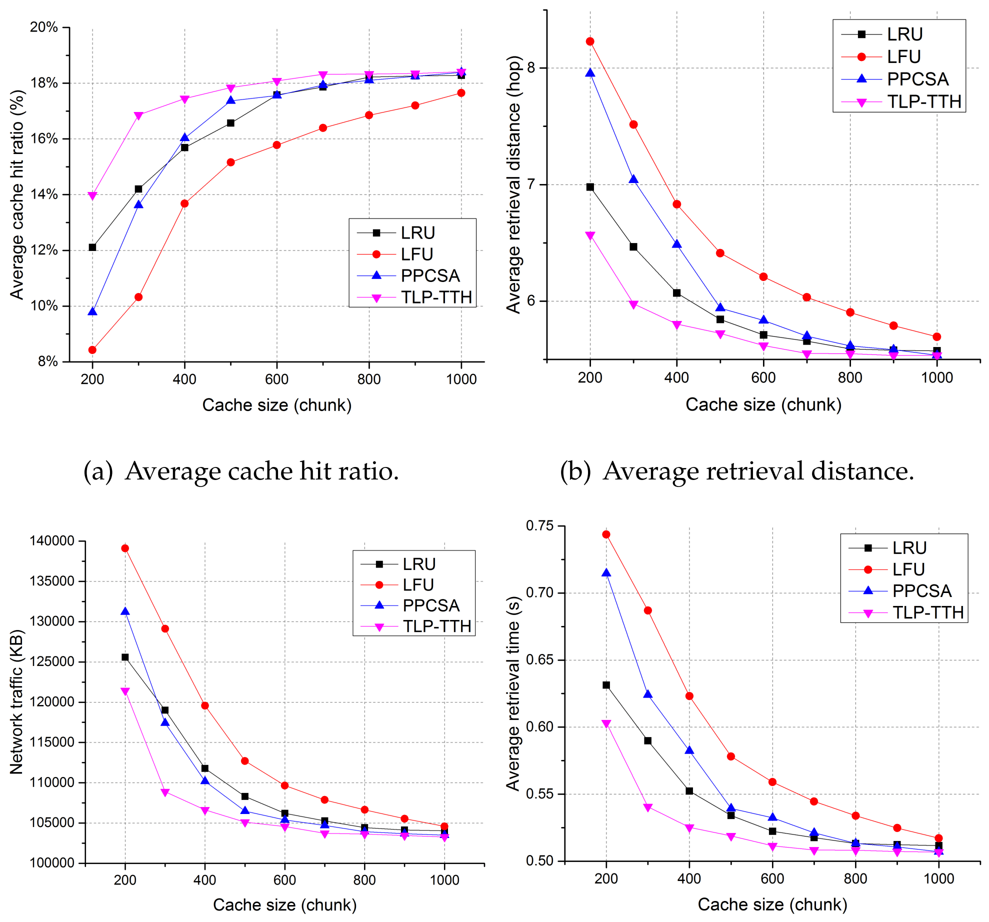
FIGURE 6.7: Average cache hit ratio in different network domains of hybrid topology.

Figure 6.7 shows the average cache hit ratio in different domains of the hybrid-topology network, which indicates that TLP-TTH increases the cache hit ratio in both edge and core networks and is not affected by the complexity of the network topology.

6.4.3 Scenario 3: Hybrid Topology with Viewer Departure

In the previous two scenarios, I assumed that all users watched the video completely without interruption. According to the observation of study [84], I aim to further evaluate the performance of the proposal close to the reality where viewers may stop watching video before its end.

In this scenario, I adopted the same topology and evaluation configuration of scenario 2. The only difference is the viewers may stop requesting video chunks while they are watching. I stipulated that the viewers departure time is the time period from the viewers start to request the video till they stop. According to the observation of study [106], a Pareto-distribution (with mean between 54% to 61% of the total video length) fits well with the measured viewer departure time. Therefore, I assumed that the viewers departure time follows the Pareto-distribution and the mean is 57% of the video duration.



(a) Average cache hit ratio. (b) Average retrieval distance.

(c) Network traffic. (d) Average retrieval time.

FIGURE 6.8: Hybrid topology with viewer departure.

The results are presented in Figure 6.8. Due to the viewer departure, all cache replacement strategies reduce their performance compared with Scenario 2. However, TLP-TTH still outperform other three cache replacement strategies. Although the performance improvement is limited in this scenario, I observe that among the four algorithms, TLP-TTH provides the highest average cache hit ratio, shortest average delivery distance (average hop count), least network traffic as well as the average retrieval time. I speculate that, TLP-TTH always assigns new TTH value to the most recently requested chunk and its subsequence chunks, and these TTH value will be updated frequently by new TTH values while this video been requested, thus, these "hot" chunks will not be replaced easily. Besides, when viewer stop requesting, this update will be stoped, and the cache size allocation algorithm will keep reducing the cache size allocated to this video. Therefore, TLP-TTH strategy will give the popular chunk and the chunks of popular video more chance to survive in CS.

6.4.4 Scenario 4: File-level Popularity Variation

In reality, the file-level popularity of videos does not remain constant, and new popular videos are requested explosively at the beginning periods when they are uploaded. Therefore, in this scenario, I aim to test the responsiveness of TLP-TTH to the variation of global file-level popularity.

The simulation configuration of scenario 4 is mostly the same as that of scenario 2. The difference is that half of the viewers who are interested in the most popular videos independently start sending their interest packets from time 20s. Still, the intervals between the starting times of viewing follow an exponential distribution. In other words, I simulate that at time 20s, several top popular videos are uploaded and start being requested, in order to evaluate the response sensitivity of cache replacement strategies to this variation of file-level popularity.

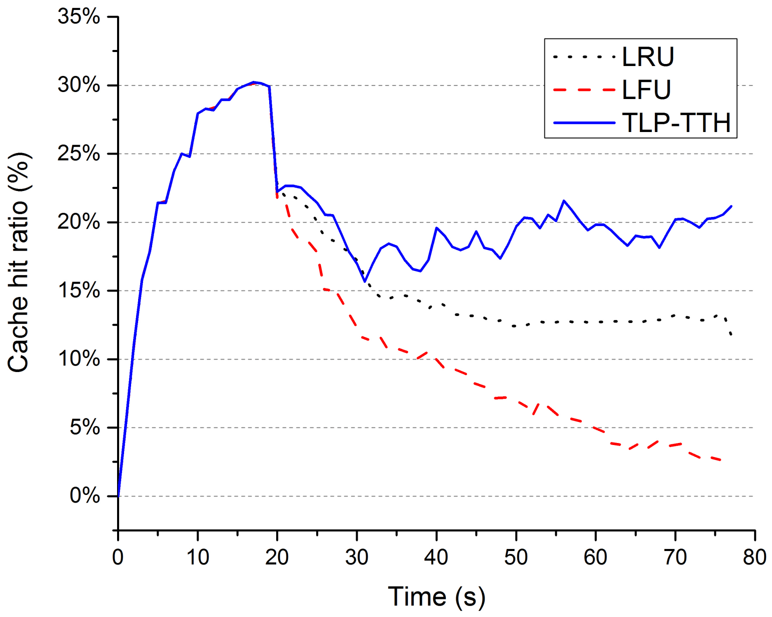


FIGURE 6.9: Average cache hit ratio in real time.

The results shown in Figure 6.9 reveal that cache strategies show different responses to the file-level popularity variation. Before 20s, the average cache hit ratios of TLP-TTH, LRU, and LFU are the same, because the cache is still available for the limited number of requests at the very edge of the network connected to the viewers. Since the popular videos are uploaded at time 20s, which generates a heavy load of content delivery for these popular videos, the cache strategies show different performance. The cache hit ratio of LFU keeps decreasing until around time 65s, and the cache hit ratio is reduced by 26%. LRU shows better responsiveness. It takes 20 seconds (from 20s to 40s) to adjust the cached chunks in the CS, and the reduction of the cache hit ratio is around 17%. TLP-TTH only needs 10 seconds to adjust the CS (from 20s to 30s) and the cache hit ratio starts increasing after 30s. Further, I believe that this increase relies on the cache size allocation algorithm that can rapidly adjust the cache resources allotted to each video in order to meet the variation of file-level popularity.

6.5 Summery

I have proposed TLP-TTH cache replacement strategy for video delivery over NDN, which consists of an evictee selection algorithm and a cache size allocation algorithm. The cache replacement strategy takes chunk-level access probability into consideration and is customized for video streaming by carefully selecting evictees that have low access probability of being requested in the future. Further, the cache space allocated to each video file is adjusted rapidly and accurately by TLP-TTH on the basis of the real time monitored file-level popularity at each NDN router. Four scenarios have been considered in the simulation to test the performance of the TLP-TTH scheme. In both the hierarchical topology and the hybrid topology, the average cache hit ratio and the average hop count are improved, which implied that the unnecessary network traffic and average content retrieval time are significantly reduced. Upon observation, I found that TLP-TTH increases the cache hit ratio in both edge and core network domains and is not affected by the complexity of the network topology. Moreover, because of the effectiveness of the cache size allocation algorithm, TLP-TTH responds sensitively to the variation of file-level popularity in order to maintain high cache utilization efficiency.

Chapter 7

RXI Cache Replacement Strategy

Since the content delivery unit over ICN has shifted from files to the segments of a file named chunks, solely using either file-level or chunk-level request probability is insufficient for ICN cache management. In this chapter, Request Expectation Index (RXI)[1] based cache replacement algorithm for streaming content delivery is proposed. In this algorithm, RXI is introduced to serve as a fine-grained and unified estimation criteria of possible future request probability for cached chunks. RXI is customized for streaming content delivery by adopting both file-level and chunk-level request probability and considering the dynamically varied request status at each route as well. Compared to prior work, the proposed algorithm evicts the chunk with the minimum expectation of future request to maintain a high cache utilization. Additionally, simulation results demonstrate that the RXI-based algorithm can remarkably enhance the streaming content delivery performance and can be deployed in complex network scenarios. The proposed results validate that, by taking fine-grained request probability and request status into consideration, the customized in-network caching algorithm can improve the ICN streaming content delivery performance by high cache utilization, fast content delivery, and lower network traffic.

[1]This work has been published in Future Internet MDPI, 2017, as the 2nd paper listed in the list of publications.

7.1 Introduction

In-network caching specifically for streaming delivery (e.g., video delivery, audio broadcasting and file distribution) over ICN architectures raises new challenges. Firstly, the cached object has a different granule size. This variation leads to inadequacy in simply extending the existing file-oriented caching research results to ICN caching studies, such as the popularity of cached objects and the request feature [49]. Additionally, for streaming content delivery over ICN, the chunk request sequence for a file is well ordered rather than generated independently. Thus, the access probability among chunks in the same file, which I call chunk-level internal probability, is traceable and calculable once the file is requested. In the case of streaming content delivery over ICN, the correlation in the occurrences of requests for chunks in the same file needs to be considered. A cache replacement algorithm for layered video content is proposed [74]. However, the experiment is conducted on a straightforward and linear network with five nodes, thus the scalability of this proposal is unclear. Two-level popularity oriented time-to-hold policy (TLP-TTH) [107] also adopts the two levels of popularity and future requests prediction. Nevertheless, the chunk-level internal probability is not included, which may further improve the streaming content delivery performance.

In this study, to further improve the ICN streaming content delivery performance of previous studies, I propose the RXI-based cache replacement algorithm, which aims to enhance cache utilization and to reduce the content retrieval time and unnecessary network traffic. RXI considers both the file-level and chunk-level internal request probability and estimates the possible future request probability according to the dynamically varied request status at each ICN router. Subsequently, the RXI-based algorithm evicts the cached chunk that has the smallest request expectation in the future in order to keep a high cache utilization. Moreover, the RXI based algorithm avoids the additional management overhead, such as messages or control packet transmission, which increases the network burden to some extent. I have evaluated the proposed algorithm compared with LRU, LFU, and TLP-TTH in both a hierarchical topology and a hybrid topology. The results indicate that, in both scenarios, the RXI-based algorithm improves several aspects of network performance and can be deployed in complex network scenarios.

7.2 Scenario Description and Assumptions

7.2.1 In-network caching for streaming content delivery scenarios

The in-networking caching for streaming content delivery is illustrated in Figure 7.1. Once the ICN router receives a request, the cache space is traversed according to the chunk name carried by the request to locate the target chunk. If the target chunk exists in the cache space, it is forwarded to the client directly, which results in a cache hit. Otherwise, according to the forwarding strategies, the request is forwarded to other routers, which leads to a cache miss in this router. Thus, every ICN router in the network is able to act as a content producer for returning the data chunk directly. For the ICN router receiving a data chunk, whether to cache is determined according to a cache decision algorithm, and if the cache space is exhausted, the cache replacement process will be triggered. The chunk selected by a cache replacement algorithm is evicted to make room for incoming chunks.

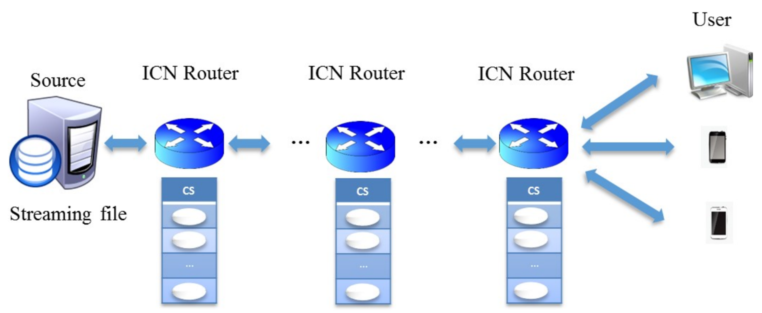


FIGURE 7.1: In-network caching for a streaming content delivery scenario

An in-network cache replacement algorithm is designed not only to identify an evictee to make room for new chunks but also to rationally manage the cache space globally to enhance the network performance in terms of the cache hit ratio, chunk delivery distance, retrieval time, etc.

7.2.2 Definitions and assumptions

I make the following assumptions, and the notation is given in Table 7.1.

1. The name of the streaming chunk in ICN is abstracted as two components: the streaming file name i and the chunk sequence number j, which can be used to distinguish different chunks of the file (e.g., */3/9*).
2. The chunk sequence number j is ordered based on the streaming sequence (e.g., the sequence number of the first chunk is 1, the second is 2, etc.).
3. Accordingly, a streaming file f_i consists of a set of chunks $\{C_{i,1}, C_{i,2}, ..., C_{i,j}...\}$, which are ordered by the sequence number j, and each chunk $C_{i,j}$ is assumed to have the same chunk size.
4. I assume that the streaming chunks are requested in sequence starting with the first chunk.

5. The request duration of f_i (the period between the start and end of one continuous user request for file f_i) follows the distribution $Q(x)$. For example, in the case of video, viewers may quit the session before the end of the video.

TABLE 7.1: Notation.

$P_f(i)$	File-level request probability of file f_i among different files at an ICN router.
$REQ_f(i)$	Local request ratio of file f_i measured at each ICN router.
δt	The time unit of the $REQ_f(i)$ measurement, e.g., $\delta t = 1s$ denotes that $REQ_f(i)$ is recalculated each second.
$P_{f_i}(j)$	Chunk-level internal request probability for chunk $C_{i,j}$, which refers to the request probability for chunk $C_{i,j}$ among all other chunks within streaming file f_i.
$IRXI_{f_i}(j)$	Internal request expectation index (IRXI) of chunk $C_{i,j}$.
$RXI(i,j)$	Request expectation index (RXI) of chunk $C_{i,j}$.
$Q(x)$	Request duration distribution.
$q(x)$	Probability density function of $Q(x)$.
ΔT_i	Average request interval of consecutive chunks in file f_i.
M_i	Total chunk number of file f_i, e.g., for chunk $C_{i,j}$, $j \in \{1, ..., M_i\}$.
n	Total number of chunks cached in a router.

7.3 Request Expectation Index Based Cache Replacement Algorithm

7.3.1 Request expectation index

$RXI(i,j)$ is the request expectation index of chunk $C_{i,j}$, which estimates the expectation of chunk $C_{i,j}$ being requested in the future. $RXI(i,j)$ is calculated according to two levels of the request probability and the recent request status, as shown in Equation 7.1:

$$RXI(i,j) = P_f(i) \times IRXI_{f_i}(j). \tag{7.1}$$

$P_f(i)$ is the real-time file-level request probability of file f_i among different cached files in the ICN router, and $\sum_{i=1}^{N} P_f(i) = 1$. $IRXI_{f_i}(j)$ indicates the internal request expectation index of chunk $C_{i,j}$, which estimates the expectation of chunk $C_{i,j}$ being requested in the future among the chunks within the file f_i. The difference between $RXI(i,j)$ and $IRXI_{f_i}(j)$ is that $RXI(i,j)$ denotes the request expectation of chunk $C_{i,j}$ among all cached chunks and $IRXI_{f_i}(j)$ only reflects the request expectation among the chunks within file f_i. For instance, assume there are two streaming files f_1 and f_2 and each file has two chunks, i.e., $f_1 = \{C_{1,1}, C_{1,2}\}$, $f_2 = \{C_{2,1}, C_{2,2}\}$. In addition, the file-level request probability of each file is given by $P_f(1) = 0.3$ and $P_f(2) = 0.7$, while, within file f_1, the request expectation of each chunk is given by $IRXI_{f_1}(1) = 0.8$ and $IRXI_{f_1}(2) = 0.2$. Then, in this ICN router, the request expectation index of chunk $C_{1,1}$ is $RXI(1,1) = P_f(1) \times IRXI_{f_1}(1) = 0.3 \times 0.8 = 0.24$.

File-level request probability

File-oriented request probability has been well established in extensive studies. For example, the request probability for web objects follows the Zipf distribution and Mandelbrot–Zipf distribution for objects delivered by P2P networks. However, the file-level probability $P_f(i)$ at an ICN router varies by time and location. Therefore, at each ICN router, the cache replacement should be adjusted based on the dynamic variation of the local file-level probability.

To achieve this goal, each router monitors the local real-time file-level access frequency to obtain request ratio $REQ_f(i)$, as shown in Figure 7.2, and the calculation of $REQ_f(i)$ is given bellow, as shown in Equation 7.2:

$$REQ_f(i) = \frac{Rec_i}{Rec}. \tag{7.2}$$

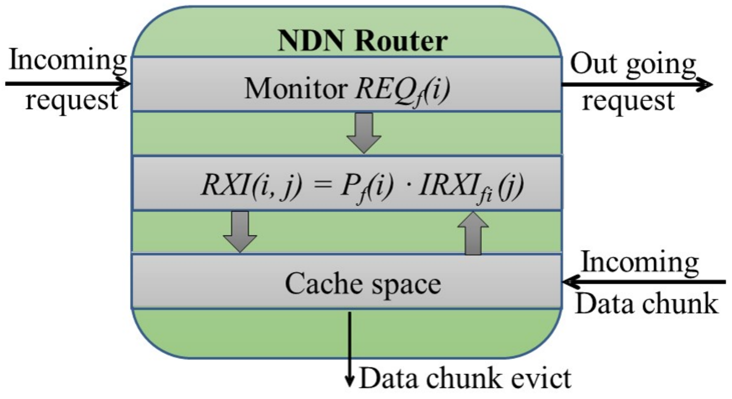


FIGURE 7.2: Monitor the request ratio $REQ_f(i)$.

Rec_i refers to the number of requests for file f_i in time period $(t, t + \delta t)$, and Rec indicates the total number of request received by this node in the same duration. Additionally, the calculation time unit δt affects the ability of

$REQ_f(i)$ to accurately reflect the variation of the local file-level request probability. $REQ_f(i)$ may be changed by the arrival of new requesters; therefore, I stipulate that the time unit for these calculations is equal to the average time interval for the arrival of a new requester. In summary, the file-level request probability $P_f(i)$ can be reasonably approximated by the measured request ratio $REQ_f(i)$, as shown in Equation 7.3:

$$P_f(i) \approx REQ_f(i). \tag{7.3}$$

Chunk-level internal request probability

For the streaming request, the user does not always request all the chunks of the file. The request probability of different chunks within a streaming file may not be equal. For instance, it is unlikely that all the viewers watch the whole video from the first chunk until the end. Thus, within a video, there is a significant request probability gap between the first chunk and the last chunk. Therefore, to accurately express this request probability difference among chunks within the same file, the chunk-level internal request probability $P_{f_i}(j)$ is introduced.

$P_{f_i}(j)$ denotes the request probability of chunk $C_{i,j}$ among the other chunks of file f_i, and $\sum_{j=1}^{M_i} P_{f_i}(j) = 1$. For a particular type of streaming file, the request duration follows a certain distribution, e.g., a Pareto distribution (with mean between 54% and 61% of the total video length) fits well for short videos which last less than 5 min, and the Wellbull distribution (with mean between 18% and 40% of the total video length) fits well for long videos [106]. Therefore, I assume that the request duration of f_i follows distribution $Q(x)$.

$P_{f_i}(j)$ denotes the inherent access frequency among chunks within a particular type of streaming file. According to the statistical characteristics of the request duration distribution $Q(x)$, $P_{f_i}(j)$ is given by Equation 7.4.

$$P_{f_i}(j) = \frac{Number\ of\ request\ for\ chunk\ C_{i,j}}{Number\ of\ total\ request\ for\ f_i}$$

$$= \frac{1 \cdot \int_{(j-1)\Delta T_i}^{j\Delta T_i} q(x)dx + 1 \cdot \int_{j\Delta T_i}^{(j+1)\Delta T_i} q(x)dx + ... + 1 \cdot \int_{(M_i-1)\Delta T_i}^{M_i\Delta T_i} q(x)dx}{1 \cdot \int_0^{\Delta T_i} q(x)dx + 2 \cdot \int_{\Delta T_i}^{2\Delta T_i} q(x)dx + ... + M_i \cdot \int_{(M_i-1)\Delta T_i}^{M_i\Delta T_i} q(x)dx}$$

$$= G_i \cdot \int_{(j-1)\Delta T_i}^{M_i\Delta T_i} q(x)dx, \tag{7.4}$$

$where$

$$G_i = \frac{1}{\sum_{k=1}^{k=M_i} k \cdot \int_{(k-1)\Delta T_i}^{k\Delta T_i} q(x)dx}.$$

In Equation 7.4, ΔT_i represents the average request interval of a user requesting two consecutive chunks for file f_i. For instance, assume ΔT_i is 1 s; accordingly, the chunk $C_{i,2}$ will be requested at time 1 s, $C_{i,3}$ will be requested at time 2 s and $C_{i,j}$ at time $(j-1)$ s, etc. Additionally, the request duration $((j-1)\Delta T_i, j\Delta T_i)$ refers that the user start to request the file from time 0 and quit between time $(j-1)\Delta T_i$ and $j\Delta T_i$. In addition, $\int_{(j-1)\Delta T_i}^{j\Delta T_i} q(x)dx$ denotes the probability that the user request duration falls within $((j-1)\Delta T_i, j\Delta T_i)$. According to the assumptions, $\int_{(j-1)\Delta T_i}^{j\Delta T_i} q(x)dx$ indicates the probability that the chunk $C_{i,j}$ is requested just before the user drops out of the session. Furthermore, a user request duration longer than time $j\Delta T_i$ indicates that chunk $C_{i,j}$ is definitely requested. Therefore, according to the $Q(x)$ distribution, the numerator of Equation 7.4 represents the number of possible requests for chunk $C_{i,j}$. Meanwhile, for the denominator, the request duration $(0, \Delta T_i)$ refers that the user quit requesting before time ΔT_i. Thus, if the request duration is in $(0, \Delta T_i)$, there is only one request for chunk $C_{i,1}$, and if it is in $(\Delta T_i, 2\Delta T_i)$, there are two requests for chunk $C_{i,1}$ and $C_{i,2}$ and so on. Thus, the denominator indicates the possible total number of requests for file f_i

based on the $Q(x)$ distribution. Since the denominator does not vary by sequence number j, I stipulate that it equals the constant $\frac{1}{G_i}$. Please note that $P_{f_i}(j)$ can be precalculated once the type of file f_i is known and $Q(x)$ for the file is determined.

Chunk-level internal request expectation index

Since I assume that users will request the streaming chunks according to the sequence number, the subsequent chunks will have a relatively high probability of being requested in the following time interval. For instance, if an ICN router receives a request for chunk $C_{2,4}$, the subsequent chunks, such as $C_{2,5}$ and $C_{2,6}$, are likely to be requested later. Therefore, if any of these subsequent chunks have already been cached, they will have higher access probability.

$IRXI_{f_i}(j)$ is estimated according to the chunk-level internal probability $P_{f_i}(j)$ and the current request status of file f_i at this router. I record $IRXI_{f_i}(j)$ for every cached chunk in the ICN router and *Update/Recover* it promptly. When a chunk is initially cached, the $IRXI_{f_i}(j)$ of the chunk is equal to the $P_{f_i}(j)$, as shown in Equation 7.5:

$$IRXI_{f_i}(j) = P_{f_i}(j). \tag{7.5}$$

$IRXI_{f_i}(j)$ is updated when the ICN router receives additional requests for the same streaming file f_i. According to the linear request feature of streaming content delivery, a received request implies that some cached subsequent chunks will likely be requested in the future; thus, the $IRXI_{f_i}(j)$ of the cached chunks is updated under the following constraints:

1. $C_{i,j}$ has been cached in the cache space of the ICN router.

2. $IRXI_{f_i}(j)$ updating only occurs when this ICN router receives a request for chunk $C_{i,g}$ of the same file f_i.

3. The sequence numbers j of the cached chunks are larger than the sequence number of the received request, i.e., $j > g$.

The $IRXI_{f_i}(j)$ value is updated on the basis of the $P_{f_i}(j)$ gap between the just-received request $C_{i,g}$ and the first chunk of file f_i, as shown in Equation 7.6:

$$\begin{aligned} IRXI_{f_i}(j) &= P_{f_i}(j) + \Delta\mu, \quad j > g, \\ where \quad \Delta\mu &= P_{f_i}(1) - P_{f_i}(g). \end{aligned} \tag{7.6}$$

$\Delta\mu$ is calculated according to the following considerations. First of all, if there are no new users requesting file f_i, the chunk $C_{i,g}$ having just been requested refers to the next subsequent chunk $C_{i,g+1}$ temporarily having the highest internal request probability. Because those chunks whose sequence number are less than or equal to g will never be requested again by the same user. Therefore, I increase $IRXI_{f_i}(g+1)$ by $P_{f_i}(1) - P_{f_i}(g)$ so that $IRXI_{f_i}(g+1)$ is the highest one currently. Secondly, the $IRXI_{f_i}(j)$ of the cached subsequent chunks will be added the same value $\Delta\mu$ so that the same request probability tendency remains compared with the chunk-level internal request probability $P_{f_i}(j)$. Finally, if there is a new user request for file f_i, which refers to the first chunk $C_{i,1}$ being requested, according to Equation 7.6, the $IRXI_{f_i}(j)$ of chunks in file f_i will return back to the initial value and equal to $P_{f_i}(j)$.

When a request for $C_{i,g}$ is received, the subsequent chunks of file f_i will likely be requested later. Therefore, the $IRXI_{f_i}(j)$ of the subsequent chunks is temporarily increased by $\Delta\mu$, that is, if any of these chunks are cached in the router, they will have high probability of being requested soon and should not be easily replaced by the cache replacement algorithm, as shown in Figure 7.3.

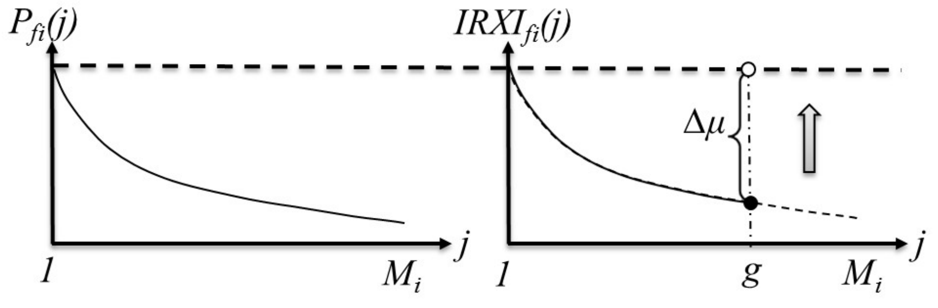


FIGURE 7.3: $IRXI_{f_i}(j)$ calculation.

The temporarily increased $IRXI_{f_i}(j)$ of the subsequent chunks should be recovered to $P_{f_i}(j)$ after being requested, and, based on Equations 7.5 and 7.6, the calculation of $IRXI_{f_i}(j)$ is summarized in Equation 7.7.

$$IRXI_{f_i}(j) = \begin{cases} P_{f_i}(j), & C_{i,j}\ is\ initially\ cached; \\ P_{f_i}(j) + \Delta\mu, & C_{i,g}\ is\ requested, (j > g); \\ P_{f_i}(j), & C_{i,j}\ is\ requested. \end{cases} \tag{7.7}$$

The Update/Recover procedure is presented in Algorithm 7.1. When an ICN router receives a request for chunk $C_{I,g}$, the cache space is traversed and the value of $IRXI_{f_i}(j)$ within file f_i is checked for an updating or recovery. According to Algorithm 7.1, the time complexity of $IRXI_{f_i}(j)$ update/recover process is $O(n)$, where n denotes the total number of chunks cached in the router.

Algorithm 7.1 $IRXI_{f_i}(j)$ Update/Recover

```
Require: $P_{f_i}(j)$, $\Delta\mu$, $C_{I,g}$
1: while All cached chunk $C_{i,j}$ in the cache space do
2:   if $i = I$ then
3:     if $j > g$ then
4:       $IRXI_{f_i}(j) \leftarrow P_{f_i}(j) + \Delta\mu$
5:     else if $j = g$ then
6:       $IRXI_{f_i}(j) \leftarrow P_{f_i}(j)$
7:     end if
8:   end if
9: end while
```

In summary, according to Equations 7.1–7.4 and 7.7, the $RXI(i,j)$ of chunk $C_{i,j}$ is given, which is the gist of the RXI algorithm.

7.3.2 RXI-Based cache replacement algorithm

In an ICN router, the chunk with the minimum $RXI(i,j)$ denotes that the chunk $C_{i,j}$ has the lowest expectation of being requested in the future. Therefore, when cache replacement occurs, the $RXI(i,j)$ of each cached chunk is calculated according to the current value of $P_f(i)$ and $IRXI_{f_i}(j)$. Subsequently, the cached chunk with the minimum $RXI(i,j)$ is evicted, as shown in Algorithm 7.2. Since the $RXI(i,j)$ of each cached chunk is calculated when cache replacement occurs, a traversal of cache space is needed for the calculations; thus, the time complexity of RXI-based cache replacement algorithm is $O(n)$.

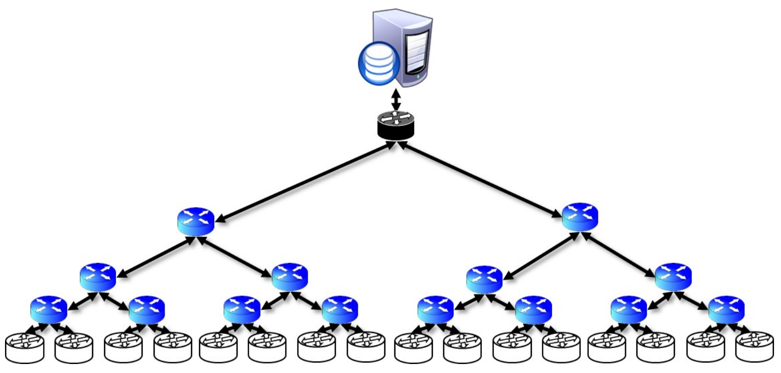
Algorithm 7.2 RXI-based Cache Replacement

```
Require: $P_f(i)$, $IRXI_{f_i}(j)$, $MinRXI$, $MinChunk$
1: $MinRXI \leftarrow 1$
2: while All cached chunk $C_{i,j}$ in the cache space do
3:     $RXI(i,j) \leftarrow P_f(i) \times IRXI_{f_i}(j)$
4:     if $MinRXI > RXI(i,j)$ then
5:         $MinRXI \leftarrow RXI(i,j)$
6:         $MinChunk \leftarrow C_{i,j}$
7:     end if
8: end while
9: Evict the chunk $MinChunk$
```

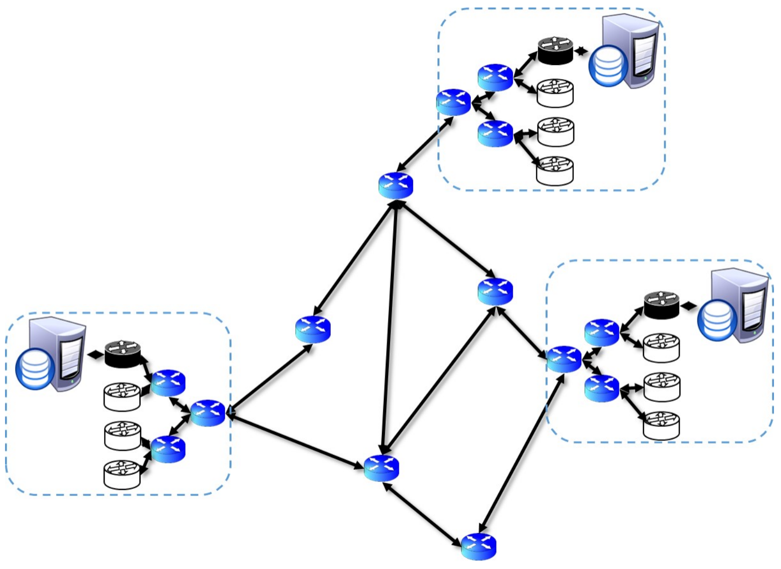
7.4 Simulations and Results

I implemented the RXI cache replacement algorithm in the ndnSIM simulator. I compared the proposed scheme with other cache replacement algorithms: LRU, LFU and TLP-TTH. To explicitly address the performance of the RXI algorithm, I adopted the default cache decision strategy named Leave Copy Everywhere (LCE) to work cooperatively with the cache replacement algorithm.

The simulation is implemented by using two topologies: a 5-level complete binary tree topology, as shown in Figure 7.4(a), and a hybrid topology consisting of a core network and access networks, as shown in Figure 7.4(b). The China Education and Research Network (CERNET) [81], which is the first nationwide education and research computer network in China, is selected as the core network, and I use a 3-level complete binary tree for the access networks.



(a)



(b)

FIGURE 7.4: Simulation topologies. (**a**) hierarchical topology; (**b**) hybrid topology.

According to the Cisco Virtual Networking Index (CVNI) project [1], the video streaming delivery has become a major consumer of network traffic and is representative of streaming transmission. I therefore customize the chunk-by-chunk video streaming request model for the simulations in both scenarios. Four metrics are adopted to evaluate the performance of the replacement strategies in each scenario:

1. *Average cache hit ratio* is used as the primary performance metric to measure the cache utilization efficiency. In the simulation, the cache hit ratio

is calculated individually by each router and represents the number of cache hits divided by the number of requests received by each router.

2. *Average retrieval distance* indicates the average transmission round-trip distance (the sum of the hops of one communication from sending a request till receiving the requested chunk) for each video chunk delivery.

3. *Network traffic* represents the sum of the data forwarded by every ICN router during the entire simulation.

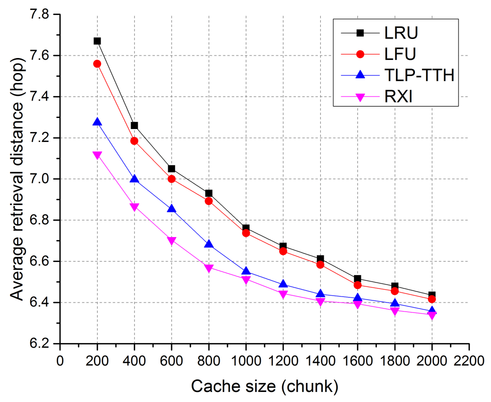
4. *Total retrieval time* denotes the sum of the retrieval time between a request being sent and the data bing received by the users.

7.4.1 Scenario 1: hierarchical topology with one producer

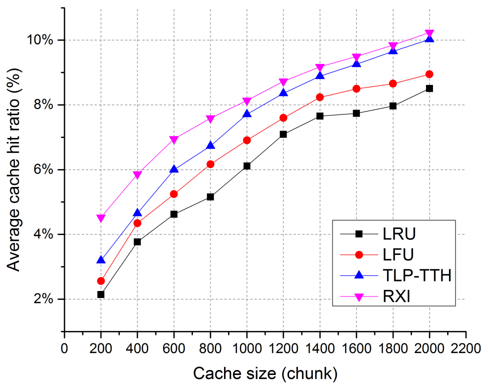
In this scenario, I evaluated the performance of the cache replacement algorithms in the 5-level complete binary tree topology. I implemented one video streaming server connected to the root router as the black router shown in Figure 7.4(a). Twenty-five video streaming files are supplied on this server, and each contains 1000 chunks that can be requested independently. Accordingly, the total chunk number is 25,000 items. With respect to the file-level popularity, studies have shown that it follows the Zipf distribution. Therefore, I implement 100 video viewers connected to the leaf routers, as represented by the white routers in Figure 7.4(a), and the popularity of requests for different video files follows a Zipf distribution (α = 0.8) [105]. In addition, these viewers request videos with given intervals that are assumed to follow the Exponential distribution with a mean of 1 s, that is, the time unit δt of the $REQ_f(i)$ measurement is set to 1 s. Additionally, the viewer requests for a video following the order of the sequence number j from the beginning of the video, but may stop before the end of the video. According to the observations in study [106], a Pareto distribution (with the mean between 54%

and 61% of the total video length) fits the measured viewer request duration. Therefore, I assume that the request duration of viewers follows the Pareto distribution with a mean of 57% of the video duration. The simulation is performed across a range of cache capacity from 200 chunks to 2000 chunks for each ICN node.

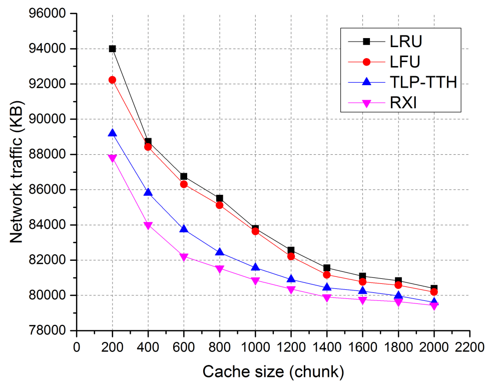
The simulation results presented in Figure 7.5(a) and 7.5(b) show significant differences among the LRU, LFU, TLP-TTH and RXI-based algorithm. The average cache hit ratio and the average hop count are calculated among each ICN router with the router cache capacity ranging from 200 chunks to 2000 chunks. Compared with TLP-TTH, the average cache hit ratio is improved from 7.44% to 8.05% by the RXI-based algorithm, thus the average cache hit ratio increase is over 8%, while TLP-TTH shows an 11% increase over LFU. Therefore, the overall performance increase is approximately 20% for the RXI-based algorithm compared with LFU and 32% compared with LRU. The RXI-based algorithm shows a 1.1% decrease in average transfer distance compared with TLP-TTH and a nearly 3.3% decrease compared with LFU. The results above show that, compared with the reference algorithms, the RXI-based algorithm enhances the utilization of the cache space; thus, viewers can acquire video chunks from closer routers. In other words, the RXI-based algorithm lightens the pressure on the original video sources. The RXI-based algorithm reduces the total network traffic and the total retrieval time compared with LRU, LFU and TLP-TTH, as shown in Figure 7.5(c) and 7.5(d) . Because of the reduction in the average delivery distance between viewers and the target content chunks, the unnecessary network traffic among intermediate routers and the average retrieval time are reduced. These reductions are further intensified by the high cache hit ratio of the RXI-based algorithm. In conclusion, to deliver the same number of video chunks, the RXI-based algorithm consumes less network resources and requires less time than LRU, LFU and TLP-TTH, and when the cache size is limited (e.g., cache size $<$ 1400 chunks), the improvement is more evident.



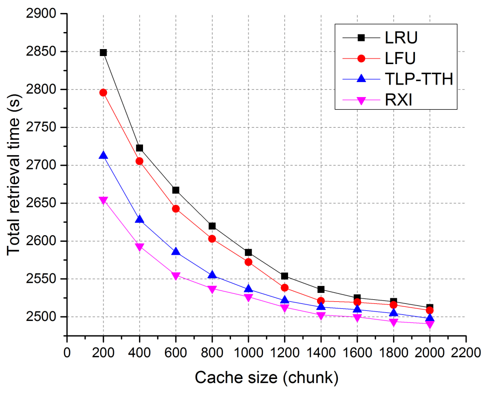
(a)



(b)



(c)



(d)

FIGURE 7.5: Simulation results of hierarchical topology. (**a**) average retrieval distance; (**b**) average cache hit ratio; (**c**) network traffic; (**d**) total retrieval time.

The performance improvements are mainly due to the following facts. Fine-grained estimation of future request probability helps the accurate prediction of future request behaviors, so that the underutilized cache capacity can be efficiently used. For instance, both the TLP-TTH and RXI-based algorithm adopt two-levels of the request popularity for cache replacement, and both algorithms significantly improve the network performance compared with LRU and LFU, which only take the simple chunk-by-chunk request probability into account, as confirmed in all four figures of Figure 7.5. In addition, unlike TLP-TTH, the RXI-based algorithm includes the chunk-level internal request probability and introduces a unified estimation criteria of possible future request probability for all cached chunks, which may belong to different files and be located in different positions in the file. Therefore, by further considering the internal relationships among chunks within a file, the estimation of future request probability is more accurate, so that the network performance is further improved by the RXI-based algorithm, as shown in Figure 7.5. Nevertheless, accurate estimation of request probability requires more calculation resources. Due to the simple principle of LRU and LFU, the eviction time complexity can achieve $O(1)$. In contrast, the eviction time complexity of TLP-TTH and RXI-based algorithm is higher, but still able to attain linear time complexity $O(n)$.

In addition, I am curious about how the RXI-based algorithm performs in different domains of a network. Therefore, I recalculate the cache hit ratio of scenario 1 (cache size = 800 chunks) in different network regions. One interesting observation is that, compared with the other algorithms, the RXI-based algorithm increases the cache hit ratio at the edge of the network and at the intermediate routers allocated at the core network, as shown in Figure 7.6. The improvement in the intermediate network is mostly due to the accurate and independent estimation of the future request probability of the cached chunks at each router, and the eviction of the chunk that has the lowest expectation of been requested later.

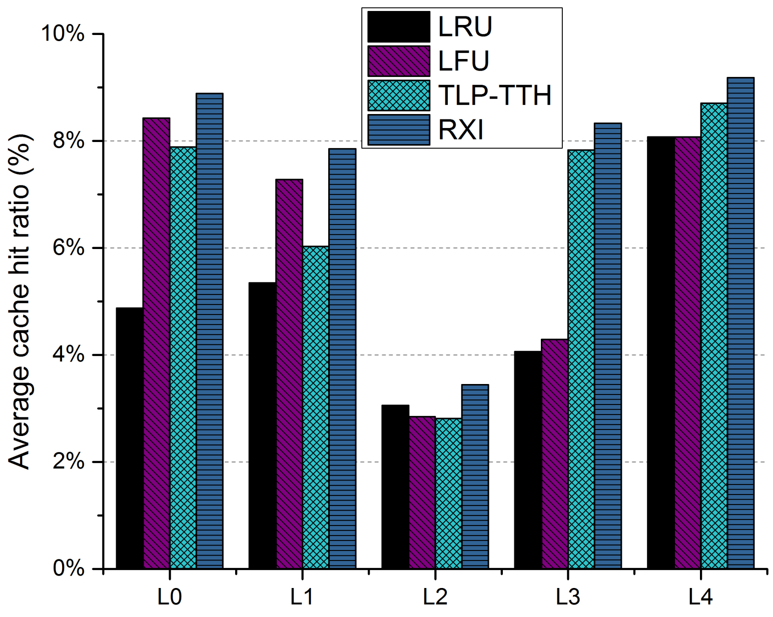


FIGURE 7.6: Average cache hit ratio in different levels of hierarchical topology.

7.4.2 Scenario 2: hybrid topology with multiple producers

In this scenario, I aim to present the network performance of cache schemes on a complex hybrid network topology. To evaluate the performance in a realistic scenario, video servers are connected to the edges of the access networks. I place three streaming sources that are connected to the black routers, and 100 video viewers are connected to other leaf nodes (white routers) in each access network, as shown in Figure 7.4(b). I randomly place 25 video streaming files in these streaming sources so that the viewers may request these videos from other access networks. With the exception of the topology difference, all of the configurations are the same as in scenario 1.

In the complex network topology and request model, the performance improvement of the RXI-based algorithm is still good, as shown in Figure 7.7. Since the average cache hit ratio and the average hop count are significantly improved by the RXI-based algorithm compared with the other three strategies, unnecessary network traffic and the content retrieval time are reduced.

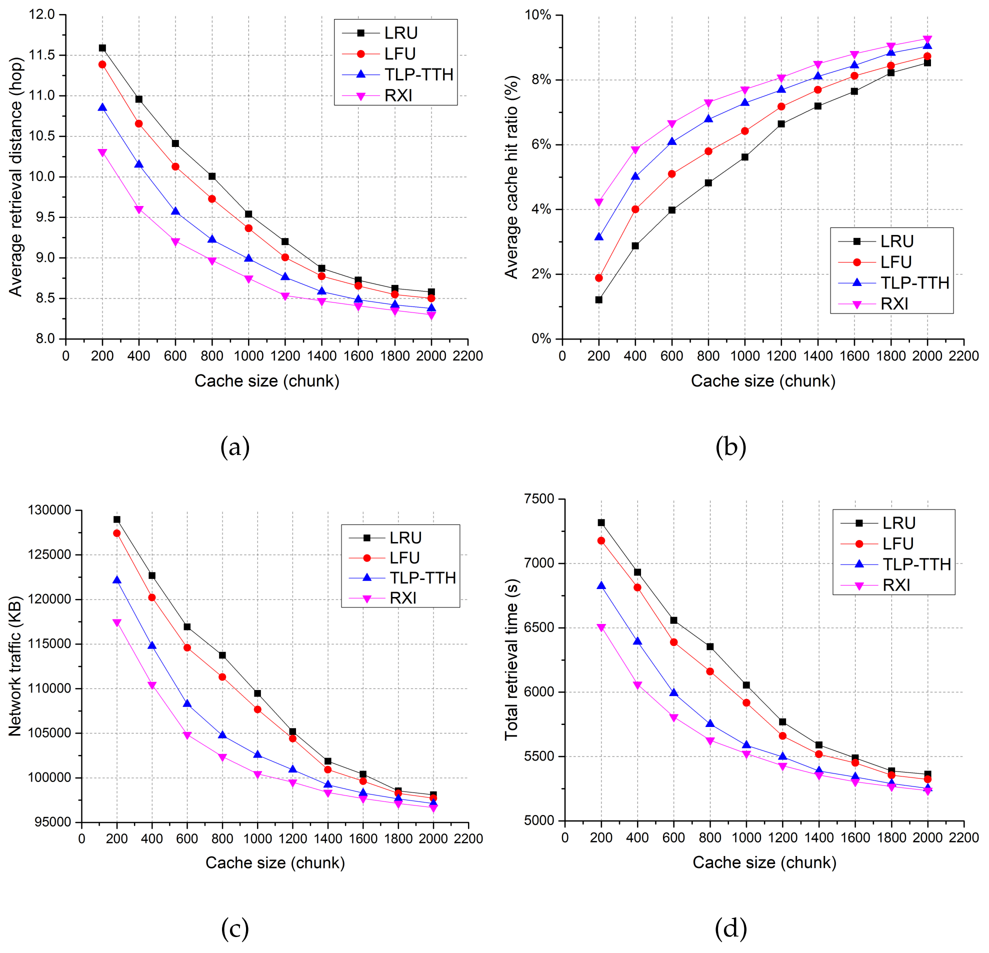


FIGURE 7.7: Simulation results of hybrid topology. (**a**) average retrieval distance; (**b**) average cache hit ratio; (**c**) network traffic; (**d**) total retrieval time.

Because the $RXI(i, j)$ is calculated independently at each router based on real-time observations and current streaming request status, the complexity of the network topology and the request model does not affect the performance of the RXI-based algorithm. In addition, the RXI-based algorithm can also avoid additional management overhead, such as messages or control packets transmission, which may obviously increase the network burden in the complex network topology. On the basis of these results, I can conclude that the RXI-based algorithm can be deployed in complex network scenarios.

7.5 Summery

In this chapter, the in-network caching management algorithm over ICN is investigated. RXI-based caching replacement algorithm customized for streaming delivery over ICN is proposed. I have introduced RXI to offer a unified estimation criteria of possible future request probability for cached chunks. RXI adopts both file-level and chunk-level internal request probability and is estimated according to the dynamically varied request status at each ICN router. I have evaluated the proposed algorithm in two scenarios by comparing with the LRU, LFU and TLP-TTH. The simulation results indicate that the RXI-based algorithm evicts the cached chunk with the minimum RXI in order to maintain high cache utilization, and improves the ICN network performance in several aspects. In addition, the RXI-based algorithm can also avoid additional management overhead, such as control packets transmission. Thus, the proposed algorithm can be deployed in complex network scenarios, as proved by the results. In conclusion, by taking fine-grained request probability and dynamically varied request status into consideration, the customized in-network caching management algorithm specifically for streaming delivery can further improve the ICN network performance.

Chapter 8

Conclusion and Future Works

Cache management strategy that only considers file-oriented popularity results in inefficient utilization of cache resources. Thus by considering the fine-grained popularity leads to precise management of the cache space for better performance. For streaming delivery scenario, the cache management strategy needs to be customized to meet the liner streaming request characteristics in order to improve the network performance.

I have proposed 6 cache management strategies for streaming content delivery over ICN, which include cache decision strategies and cache replacement strategies. Simulation results show that the proposed cache management strategies improve the streaming content delivery over ICN dramatically.

Finally, energy-efficient cache management is still an open issue for content delivery over ICN, and in order to prepare for the explosive growth of the current streaming traffic, the energy issue needs to be considered. Further, to improve the user-perceived quality of videos, dynamic adaptation streaming over HTTP (DASH) was proposed. Thus, an efficient management of the cached videos with DASH is an interesting challenge.

Bibliography

[1] Cisco, "Cisco Visual Networking Index : Global Mobile Data Traffic Forecast Update , 2013 – 2018," 2015.

[2] Cicso, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2015–2020 White Paper," Cisco Public, 2016. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html.

[3] M. F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and B. Mathieu, "A survey of naming and routing in information-centric networks," IEEE Communications Magazine, vol. 50, no. 12. pp. 44–53, 2012.

[4] G. Xylomenos et al., "A Survey of information-centric networking research," IEEE Communications Surveys and Tutorials, vol. 16, no. 2. pp. 1024–1049, 2014.

[5] N. Choi, K. Guan, D. C. Kilper, and G. Atkinson, "In-network caching effect on optimal energy consumption in content-centric networking," in IEEE International Conference on Communications, 2012, pp. 2889–2894.

[6] M. Zhang, H. Luo, and H. Zhang, "A survey of caching mechanisms in information-centric networking," IEEE Commun. Surv. Tutorials, vol. 17, no. 3, pp. 1473–1499, 2015.

[7] M. Gritter and D. R. Cheriton, “An architecture for content routing support in the internet,” 3rd USENIX Symp. Internet Technol. Syst., vol. 15, no. 5, pp. 4–4, 2001.

[8] D. R. Cheriton and M. Gritter, “TRIAD: A new next-generation Internet architecture,” Comput. Sci. Dep. Stanford Univ., 2001.

[9] T. Koponen et al., “A data-oriented (and beyond) network architecture,” ACM SIGCOMM Comput. Commun. Rev., vol. 37, no. 4, p. 181, 2007.

[10] N. Fotiou, P. Nikander, D. Trossen, and G. C. Polyzos, “Developing information networking further: From PSIRP to PURSUIT,” in Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, 2012, vol. 66 LNICST, pp. 1–13.

[11] Dannewitz, Christian. "Netinf: An information-centric design for the future internet." Proc. 3rd GI/ITG KuVS Workshop on The Future Internet. 2009.

[12] L. Zhang et al., “Named Data Networking (NDN) Project,” October, pp. 1–26, 2010.

[13] L. Zhang et al., “Named data networking,” ACM SIGCOMM Comput. Commun. Rev., vol. 44, no. 3, pp. 66–73, 2014.

[14] N. Niebert et al., “The way 4WARD to the creation of a future internet,” 2008 IEEE 19th Int. Symp. Pers. Indoor Mob. Radio Commun., pp. 0–4, 2008.

[15] W. Consortium, The FP7 4WARD Project, Technical report, 2008. http://www. 4ward-project. eu.

[16] T. Edwall, "Scalable & adaptive internet solutions (SAIL)," Report, 2011.

[17] M. Brunner, “Scalable & adaptive Internet solutions (SAIL),” Future Internet Assembly (FIA), Ghent, Belgium, 2010.

[18] M. D'Ambrosio, C. Dannewitz, H. Karl, and V. Vercellone, "MDHT: A Hierarchical Name Resolution Service for Information-centric Networks," Proc. ACM SIGCOMM Work. Information-centric Netw. - ICN '11, p. 7, 2011.

[19] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," Internet Math., vol. 1, no. 4, pp. 485–509, 2004.

[20] L. A. Belady, "A study of replacement algorithms for a virtual-storage computer," IBM Syst. J., vol. 5, no. 2, pp. 78–101, 1966.

[21] R. L. Mattson, J. Gecsei, D. R. Slutz, and I. L. Traiger, "Evaluation techniques for storage hierarchies," IBM Syst. J., vol. 9, no. 2, pp. 78–117, 1970.

[22] W. Vogler, "Another short proof of optimality for the MIN cache replacement algorithm," Inf. Process. Lett., vol. 106, no. 5, pp. 219–220, 2008.

[23] M. F. Arlitt,R. J. Friedrich,T. Y. Jin, "Web cache performance by applying different replacement policies to the web cache." U.S. Patent No. 6,272,598. 7 Aug. 2001.

[24] S. Romano, H. ElAarag, "A quantitative study of recency and frequency based web cache replacement strategies." Proceedings of the 11th communications and networking simulation symposium. ACM, 2008.

[25] H. R. Sadjadpour, "A new design for Information Centric Networks," 2014 48th Annu. Conf. Inf. Sci. Syst., pp. 1–6, 2014.

[26] F. M. Al-Turjman and H. S. Hassanein, "Enhanced data delivery framework for dynamic Information-Centric Networks (ICNs)," in Proceedings - Conference on Local Computer Networks, LCN, 2013, pp. 810–817.

[27] M. Vahlenkamp, F. Schneider, D. Kutscher, and J. Seedorf, "Enabling Information-Centric Networking in IP Networks Using SDN," Futur.

Networks Serv. (SDN4FNS), IEEE, pp. 1–6, 2013.

[28] G. Domingues, E. e Silva, R. Leao, and D. Menasché, "Enabling Information Centric Networks through Opportunistic Search, Routing and Caching," 31st Brazilian Symp. Comput. Networks Distrib. Syst. SBRC' 2013, 2013.

[29] G. M. Brito, P. B. Velloso, and I. M. Moraes, Information-Centric Networks: A New Paradigm for the Internet. 2013.

[30] B. J. Ko et al., "An information-centric architecture for data center networks," in Proceedings of the second edition of the ICN workshop on Information-centric networking - ICN '12, 2012, p. 79.

[31] G. Carofiglio, G. Morabito, L. Muscariello, I. Solis, and M. Varvello, "From content delivery today to information centric networking," Comput. Networks, vol. 57, no. 16, pp. 3116–3127, 2013.

[32] F. Lin, Y. Chen, J. An, and X. Zhou, "A hierarchical name system based on hybrid P2P for data-centric networks," in Proceedings - 2012 4th International Conference on Multimedia and Security, MINES 2012, 2012, pp. 632–635.

[33] V. Sourlas, P. Flegkas, and L. Tassiulas, "A novel cache aware routing scheme for Information-Centric Networks," Comput. Networks, vol. 59, pp. 44–61, 2014.

[34] A. Dabirmoghaddam, M. M. Barijough, and J. 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 - INC '14, 2014, pp. 47–56.

[35] 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 - INC '14, 2014, pp. 37–46.

[36] I. Psaras, W. K. Chai, and G. Pavlou, "In-network cache management and resource allocation for information-centric networks," IEEE Trans. Parallel Distrib. Syst., vol. 25, no. 11, pp. 2920–2931, 2014.

[37] 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 - ICN '12, 2012, p. 55.

[38] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache "less for more" in information-centric networks." International Conference on Research in Networking. Springer, Berlin, Heidelberg, 2012.

[39] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache 'less for more' in information-centric networks (extended version)," Comput. Commun., vol. 36, no. 7, pp. 758–770, 2013.

[40] F. Kocak, G. Kesidis, T. M. Pham, and S. Fdida, "The effect of caching on a model of content and access provider revenues in Information-Centric Networks," in Proceedings - SocialCom/PASSAT/BigData/EconCom/BioMedCom 2013, 2013, pp. 45–50.

[41] V. Sourlas, L. Gkatzikis, P. Flegkas, and L. Tassiulas, "Distributed cache management in information-centric networks," IEEE Trans. Netw. Serv. Manag., vol. 10, no. 3, pp. 286–299, 2013.

[42] V. Sourlas and L. Tassiulas, "Effective cache management and performance limits in information-centric networks," in 2013 International Conference on Computing, Networking and Communications, ICNC 2013, 2013, pp. 955–960.

[43] S. Saha, A. Lukyanenko, and A. Yla-Jaaski, "Cooperative caching through routing control in information-centric networks," in Proceedings - IEEE INFOCOM, 2013, pp. 100–104.

[44] F. M. Al-Turjman, A. E. Al-Fagih, and H. S. Hassanein, "A value-based cache replacement approach for Information-Centric Networks," in Proceedings - Conference on Local Computer Networks, LCN, 2013, pp. 874–881.

[45] L. Galluccio, G. Morabito, and S. Palazzo, "Caching in information-centric satellite networks," in IEEE International Conference on Communications, 2012, pp. 3306–3310.

[46] M. Dräxler and H. Karl, "Efficiency of on-path and off-path caching strategies in information centric networks," in Proceedings - 2012 IEEE Int. Conf. on Green Computing and Communications, GreenCom 2012, Conf. on Internet of Things, iThings 2012 and Conf. on Cyber, Physical and Social Computing, CPSCom 2012, 2012, pp. 581–587.

[47] X. Vasilakos, V. a Siris, G. C. Polyzos, and M. Pomonis, "Proactive selective neighbor caching for enhancing mobility support in information-centric networks," in Proceedings of the second edition of the ICN workshop on Information-centric networking - ICN '12, 2012, p. 61.

[48] Z. Ming, M. Xu, and D. Wang, "Age-based cooperative caching in Information-Centric Networks," in Proceedings - IEEE INFOCOM, 2012, pp. 268–273.

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

[50] J. M. Wang, J. Zhang, and B. Bensaou, "Intra-AS cooperative caching for content-centric networks," Proc. 3rd ACM SIGCOMM Work. Information-centric Netw. - ICN '13, p. 61, 2013.

[51] S. Hassan, Z. Aziz, K. Nisar, "On the cache performance of the information centric network." Computing, Electrical and Electronics Engineering (ICCEEE), 2013 International Conference on. IEEE, 2013.

[52] H. Y. Lee and A. Nakao, "User-assisted in-network caching in information-centric networking," Comput. Networks, vol. 57, no. 16, pp. 3142–3153, 2013.

[53] S.-Z. Yu, W.-T. Wu, Y. Gao, and W.-X. Liu, "Caching efficiency of information-centric networking," IET Networks, vol. 2, no. 2, pp. 53–62, 2013.

[54] K. Suksomboon et al., "PopCache: Cache more or less based on content popularity for information-centric networking," in Proceedings - Conference on Local Computer Networks, LCN, 2013, pp. 236–243.

[55] S. Wang, J. Bi, and J. Wu, "On performance of cache policy in information-centric networking," in 2012 21st International Conference on Computer Communications and Networks, ICCCN 2012 - Proceedings, 2012.

[56] E. Yeh, T. Ho, Y. Cui, M. Burd, R. Liu, and D. Leong, "VIP: A framework for joint dynamic forwarding and caching in named data networks," in Proceedings of the 1st international conference on Information-centric networking - INC '14, 2014, pp. 117–126.

[57] M. Gallo, B. Kauffmann, L. Muscariello, A. Simonian, and C. Tanguy, "Performance evaluation of the random replacement policy for networks of caches," Perform. Eval., vol. 72, pp. 16–36, 2014.

[58] Y. Kim and I. Yeom, "Performance analysis of in-network caching for content-centric networking," Comput. Networks, vol. 57, no. 13, pp. 2465–2482, 2013.

[59] N. C. Fofack, P. Nain, G. Neglia, and D. Towsley, "Analysis of TTL-based cache networks," Perform. Eval. Methodol. Tools (VALUETOOLS), 2012 6th Int. Conf., pp. 1–10, 2012.

[60] J. Ardelius, B. Grönvall, L. Westberg, and Å. Arvidsson, "On the effects of caching in access aggregation networks," in Proceedings of the second edition of the ICN workshop on Information-centric networking - ICN '12, 2012, p. 67.

[61] A. Detti, M. Pomposini, N. Blefari-Melazzi, and S. Salsano, "Supporting the Web with an information centric network that routes by name," Comput. Networks, vol. 56, no. 17, pp. 3705–3722, 2012.

[62] K. Katsaros, G. Xylomenos, and G. C. Polyzos, "MultiCache: An overlay architecture for information-centric networking," Comput. Networks, vol. 55, no. 4, pp. 936–947, 2011.

[63] T. Braun, V. Hilt, M. Hofmann, I. Rimac, M. Steiner, and M. Varvello, "Service-centric networking," in IEEE International Conference on Communications, 2011.

[64] A. Detti, N. Blefari-Melazzi, S. Salsano, and M. Pomposini, "CONET: A Content Centric Inter-Networking Architecture," Proc. ACM SIGCOMM Work. Information-centric Netw. - ICN '11, pp. 50–55, 2011.

[65] L. Zhao, Y. Zaki, A. Udugama, U. Toseef, C. Gorg, and A. Timm-Giel, "Open Connectivity Services for future networks," in 2011 8th International Conference and Expo on Emerging Technologies for a Smarter World, CEWIT 2011, 2011.

[66] K. Pentikousis and T. Rautio, "A multiaccess network of information," in 2010 IEEE International Symposium on "A World of Wireless, Mobile and Multimedia Networks", WoWMoM 2010 - Digital Proceedings, 2010.

[67] J. Chen, et al. "Coexist: a hybrid approach for content oriented publish/subscribe systems." Proceedings of the second edition of the ICN workshop on Information-centric networking. ACM, 2012.

[68] M. F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and B. Mathieu, "A survey of naming and routing in information-centric networks," IEEE Communications Magazine, vol. 50, no. 12. pp. 44–53, 2012.

[69] C. Fang, F. R. Yu, T. Huang, J. Liu, and Y. Liu, "A survey of green information-centric networking: Research issues and challenges," IEEE Commun. Surv. Tutorials, vol. 17, no. 3, pp. 1455–1472, 2015.

[70] G. Tyson, N. Sastry, R. Cuevas, I. Rimac, and A. Mauthe, "A survey of mobility in information-centric networks," Commun. ACM, vol. 56, no. 12, pp. 90–98, 2013.

[71] A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM: NDN simulator for NS-3," NDN, Tech. Rep. NDN-0005, pp. 1–7, 2012.

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

[73] X. Hu, C. Papadopoulos, J. Gong, and D. Massey, "Not So Cooperative Caching in Named Data Networking," in 2013 IEEE Global Communications Conference (GLOBECOM), 2013, pp. 2263–2268.

[74] J. Lee, K. Lim, and C. Yoo, "Cache replacement strategies for scalable video streaming in CCN," in 2013 19th Asia-Pacific Conference on Communications, APCC 2013, 2013, pp. 184–189.

[75] J. Matlis, "Internet2," Computerworld, vol. 40, no. 35, p. 30, 2006.

[76] Y. Li, et al. "Coordinating in-network caching in content-centric networks: Model and analysis." Distributed Computing Systems (ICDCS), 2013 IEEE 33rd International Conference on. IEEE, 2013.

[77] M. Dräxler and K. Holger, "Efficiency of on-path and off-path caching strategies in information centric networks." Green Computing and Communications (GreenCom), 2012 IEEE International Conference on. IEEE, 2012.

[78] Y. Wang, et al. "Optimal cache allocation for content-centric networking." Network Protocols (ICNP), 2013 21st IEEE International Conference on. IEEE, 2013.

[79] S. Awiphan, T. Muto, Y. Wang, Z. Su, and J. Katto, "Video streaming over content centric networking: Experimental studies on PlanetLab," in 2013 Computing, Communications and IT Applications Conference, ComComAp 2013, 2013, pp. 19–24.

[80] A. Detti, B. Ricci, and N. Blefari-Melazzi, "Peer-to-peer live adaptive video streaming for Information Centric cellular networks," in IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC, 2013, pp. 3583–3588.

[81] "The China Education and Research Network (CERNET) [on-line]" http://www.edu.cn/.

[82] "Internet2 [on-line]", http://www.internet2.edu/.

[83] "GÉANT Project [on-line]" http://www.geant.net/.

[84] P. Gill, M. F. Arlitt, Z. Li, and A. Mahanti, "Youtube traffic characterization: a view from the edge," in Proceedings of the 7th ACM SIGCOMM Internet Measurement Conference, IMC 2007, San Diego, California, USA, October 24-26, 2007, 2007, pp. 15–28.

[85] T. Wu, K. De Schepper, W. Van Leekwijck, and D. De Vleeschauwer, "Reuse time based caching policy for video streaming," in 2012 IEEE Consumer Communications and Networking Conference, CCNC'2012, 2012, pp. 89–93.

[86] M. Cha, H. Kwak, P. Rodriguez, Y. Ahn, and S. Moon, “I Tube , You Tube , Everybody Tubes : Analyzing the World ’ s Largest User Generated Content Video System,” IMC’07, Oct. 24-26, San Diego, California, USA, pp. 1–13, 2007.

[87] S. Imai, K. Leibnitz, and M. Murata, “Energy-aware cache management for content-centric networking,” in Proceedings - 27th International Conference on Advanced Information Networking and Applications Workshops, WAINA 2013, 2013, pp. 1623–1629.

[88] T. Hasegawa, Y. Nakai, K. Ohsugi, J. Takemasa, Y. Koizumi, and I. Psaras, “Empirically modeling how a multicore software ICN router and an ICN network consume power,” Proc. 1st Int. Conf. Information-centric Netw. - INC ’14, pp. 157–166, 2014.

[89] S. K. Fayazbakhsh et al., “Less Pain, Most of the Gain: Incrementally Deployable ICN,” Proc. ACM SIGCOMM, vol. 43, no. 4, p. 147, 2013.

[90] L. Wang, S. Bayhan, and J. Kangasharju, “Effects of Cooperation Policy and Network Topology on Performance of In-Network Caching.”

[91] Y. Thomas, G. Xylomenos, C. Tsilopoulos, and G. C. Polyzos, “Object-Oriented Packet Caching for ICN,” Proc. 2nd Int. Conf. Information-Centric Netw. - ICN ’15, pp. 89–98, 2015.

[92] S. Podlipnig and L. Böszörmenyi, “A survey of Web cache replacement strategies,” ACM Comput. Surv., vol. 35, no. 4, pp. 374–398, 2003.

[93] Xu, Jun, Mukesh Singhal, and Joanne Degroat. "A novel cache architecture to support layer-four packet classification at memory access speeds." INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE. Vol. 3. IEEE, 2000.

[94] J. (Jim) Xu, M. Singhal, and J. DeGroat, “A Novel Cache Architecture To Support Layer-four Packet Classification At Memory Access Speeds,” Proc. IEEE Int. Conf. Comput. Commun., vol. 0, no. c, pp. 1445–1454, 2000.

[95] A. A. Khandekar and S. B. Mane, “Analyzing different cache replacement policies on cloud,” in 2015 International Conference on Industrial Instrumentation and Control, ICIC 2015, 2015, pp. 709–712.

[96] N. Megiddo and D. S. Modha, “Outperforming LRU with an adaptive replacement cache algorithm,” Computer (Long. Beach. Calif)., vol. 37, no. 4, pp. 58–65, 2004.

[97] K. Y. Wong, “Web cache replacement policies: A pragmatic approach,” IEEE Netw., vol. 20, no. 1, pp. 28–34, 2006.

[98] P. Sridama, P. Somchai , and P. Nalinpat. "Web cache replacement with the repairable LRU algorithm." International Review on Computers and Software (IRECOS) 10.6 (2015): 620-627.

[99] C. Fang, T. Huang, J. Liu, J. Y. Chen, and Y. J. Liu, “Fast convergence caching replacement algorithm based on dynamic classification for content-centric networks,” J. China Univ. Posts Telecommun., vol. 20, no. 5, pp. 45–50, 2013.

[100] Fricker, Christine, Philippe Robert, and James Roberts. "A versatile and accurate approximation for LRU cache performance." Proceedings of the 24th International Teletraffic Congress. International Teletraffic Congress, pp. 1–16, 2012.

[101] H. Che, Y. Tung, and Z. Wang, “Hierarchical Web caching systems: modeling, design and experimental results,” IEEE J. Sel. Areas Commun., vol. 20, no. 7, pp. 1305–1314, 2002.

[102] S. Traverso et al., "Characterizing Reference Locality in the WWW," Parallel Distrib. Inf. Syst., vol. 43, no. 5, pp. 6–12, 2012.

[103] M. Ahmed, S. Traverso, M. Garetto, P. Giaccone, E. Leonardi, and S. Niccolini, "Temporal Locality in Today's Content Caching: Why it Matters and How to Model it," 2013.

[104] A. Brodersen, S. Scellato, and M. Wattenhofer, "YouTube Around the World : Geographic Popularity of Videos," Proc. 21st Int. Conf. World Wide Web, pp. 241–250, 2012.

[105] D. Rossi and G. Rossini, "Caching performance of content centric networks under multi-path routing (and more)," Relatório técnico, Telecom ParisTech, 2011.

[106] C. P. Costa et al., "Analyzing client interactivity in streaming media," Proc. 13th Conf. World Wide Web - WWW '04, p. 534, 2004.

[107] H. Li and H. Nakazato, "Two-level popularity-oriented cache replacement policy for video delivery over CCN," IEICE Trans. Commun., vol. E99B, no. 12, pp. 2532–2540, 2016.

[108] 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.

Appendix A

List of Publications

1. H. Li and H. Nakazato, "Two-level popularity-oriented cache replacement policy for video delivery over CCN," IEICE Trans. Commun., vol. E99B, no. 12, pp. 2532–2540, 2016.

2. H. Li, H. Nakazato, and S. H. Ahmed, "Request expectation index based cache replacement algorithm for streaming content delivery over ICN," Futur. Internet, vol. 9, no. 4, 2017.

3. H. Li, H. Nakazato, A. Detti, and N. B. Melazzi, "Popularity Proportional Cache Size Allocation policy for video delivery on CCN," in 2015 European Conference on Networks and Communications, EuCNC 2015, 2015, pp. 434–438.

4. H. Li and H. Nakazato, "Time to Hold (TTH): An Optimal Cache Replacement Policy for Video Delivery on CCN." IEICE Technical report. CS. 2014 114.289: 69-74.

5. H. Li and H. Nakazato, "Dynamic File-Level Popularity based Cache Management Scheme for Video Delivery on CCN." IEICE General Conference, Advanced Technologies in the Design, Management and Control for Future Innovative Communication Network, 2015.

6. H. Li and H. Nakazato, "A Popularity Based Cache Decision Policy of Content-Centric Networking". Proceedings of the IEICE General Conference,

BS-1. Future Network Technologies for Advanced Information and Communications Society, 2014.