ABSTRACT

FANG, XIAOZHOU. Neighbor Graph Based Proactive Caching for Seamless Handover in Content-Centric Network. (Under the direction of Dr. Wenye Wang.)

Content-Centric Network (CCN) have recently been considered as a promising architecture for the next-generation Internet. In CCN, by replacing server IP addresses with content names in the requests, all nodes are allowed to understand and process the requests via in-network caching, which is expected to avoid network bandwidth wasting and reduce the delay of service response simultaneously. In addition, routed-by-name feature in CCN solves the mobility problem in current IP network because content delivery is performed in a connectionless manner. However, the caching performance in CCN has not been well investigated in the literature, especially from the perspective of caching efficiency. Moreover, the problem of handover latency generated by retransmission has not been addressed in CCN. Handover latency needs to be minimized since it might introduce serious QoS degradation to time-sensitive applications.

In this thesis, we aim at understanding and improving the in-network caching performance in both static and mobile CCN scenarios. Because the performance measurement in static and mobile CCN scenarios are different, we adopt an application-specific methodology to model and evaluate the caching performance in CCN. In static CCN scenario, we focus on the caching efficiency improvement for on-demand contents whose requests traffic follows Zipf-like distribution; while in mobile CCN scenario, we focus on reducing the handover latency for real-time contents which are more time-sensitive.

In particular, we first conduct a comprehensive study in our mobile CCN simulation platform $MCSim$ in various CCN scenarios which are thoroughly discussed in our CCN parameter setting survey. We identify the issues in existing caching policies and offers instrumental guidance into efficient caching policy design in CCN. Second, we propose the $CachePop$, a popularity-oriented caching decision policy, based on the observation in our simulation study to enhance the caching efficiency for on-demand contents. We verify the performance of $CachePop$ through $MCsim$ and the results show that $CachePop$ solves the caching efficiency problem and avoids caching pollution. Finally, to address the handover latency problem, we optimize $CachePop$ by adding neighbor graph and proactive caching routing mechanisms and propose $NG\text{-}CachePop$ policy. Evaluation results show that $NG\text{-}CachePop$ achieves seamless handover for over 93.3% of retransmission requests in a general mobile scenario.

The work in this thesis advances our understanding of caching performance associated with emerging Content-Centric Network. Our $NG\text{-}CachePop$ policy significantly improves the in-network caching efficiency and solve the handover latency problem in mobile CCN scenarios.

Neighbor Graph Based Proactive Caching for Seamless Handover in Content-Centric Network

by
Xiaozhou Fang

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Computer Engineering

Raleigh, North Carolina

2013

APPROVED BY:

Dr. Do Young Eun	Dr. Khaled Harfoush

Dr. Wenye Wang
Chair of Advisory Committee

BIOGRAPHY

Xiaozhou Fang earned his dual Bachelor's degree in Telecommunication Engineering with Management from Beijing University of Posts and Telecommunications and Queen Mary, University of London in 2011. He started his Master's study and Ph.D. study in the Department of Electrical and Computer Engineering, North Carolina State University in August 2011 and 2013, respectively. His research focuses on Content Centric Network modeling, smartphone applications and computing.

ACKNOWLEDGEMENTS

Foremost, I would like to express my deepest gratitude to my advisor, Dr. Wenye Wang. for her excellent guidance, caring, patience, and providing me with an excellent atmosphere for doing research. I believe that the knowledge and skills that she has imparted to me are far beyond this thesis. I would like to thank my committee members, Dr. Do Young Eun and Dr. Khaled Harfoush, for their valuable feedback and comments, which significantly improved the quality of my research.

I would also like to give my thanks to my fellow labmates for their help during my Master thesis research: Lei Sun, Zhuo Lv, Yujin Li, Jianhua Zhang, Mingkui Wei, Huan Luo, Sigit Pambudi and Wei Fu.

Finally, I would like to give my special thanks to my parents. They were always supporting me and encouraging me with their best wishes.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Since the first day of the Internet that been built, the main concern of its designers was to connect resources and to disseminate information [1]. It was considered as a medium for collaboration and interaction between individuals and their computers without regard for geographic location. After decades of revolution, the dominant communication protocol TCP/IP in the Internet has been widely deployed for facilitating ubiquitous inter-connectivity. However, work [2] observes that the network paradigm, which used to be user-to-user decades ago, has been shifted to user-to-content. This is because the major use of Internet today is popular content retrieval rather than conversation between endpoints. Still, most of Internet mechanisms are optimized for addressing end points rather than catering for the location-independent content.

In order to handle the Internet traffic change, the research community proposed the concept of Information-Centric Networking (ICN). The goal of ICN is to provide a network infrastructure service that is able to accommodate todays Internet use, in particular popular content dissemination and mobility support, and more resilient to disruptions and failures [3]. ICN approach becomes a mainstream research topic, and in response to this, researchers proposed several projects such as Content-Centric Networking (CCN) [4], Data-Oriented Network Architecture (DONA) [5], Network of Information (NetInf) [6], Publish-Subscribe Internet Routing Paradigm (PSIRP) [7], Content-Aware Networks (COMET) [8] etc., that make the Internet more data or content-centric.

Among these projects, Content-Centric Networking [4] is one the most promising design and it is considered as the next generation of Internet. The Internet in CCN is moved away from its current reliance on purely end-to-end connection on to the accessing of contents that users are really care about. In CCN, the communication is based on user-driven model: user can directly send the request for content with content name without building the connection with server using location information. By moving the Internet paradigm from location-dependent to content-dependent, CCN provides lots of beneficial properties to improve the content dis-

semination. First of all, CCN targets general infrastructure that provides in-network caching so that content is distributed in a scalable and cost-efficient manner. As current Internet traffic are mostly used for high volume contents (e.g., music, photos and HD movies) dissemination [2], in-network caching is expected to reduce redundant transmissions over the same path, decrease the delay of service requests, therefore enhance user-perceived quality of experience. Secondly, CCN supports mobility and intermittent connectivity since the content delivery process is performed in a connectionless manner. Therefore, users can easily send and receive content from multiple locations, mobile devices, and diverse networks. Thirdly, CCN provides better security for content since it is built into content itself, so that integrity and trust are properties of the content rather than being properties of the connection over which the content travels. In addition, CCN uses multi-path routing and Pending Interest Table (PIT) that allow multiple sources for contents and avoid routing loops respectively.

Recognizing the great performance and robustness of CCN, numerous researches have started to work on this area. Bengt et al., in [3] provide a detailed comparison between CCN with other proposed ICN architectures (e.g., DONA, PSIRP, NetInf) with respect to their properties and design choices. Choi et al., in [9] offer an overview of CCN on naming, routing and caching perspectives. Also there are other works focusing on specific research areas of CCN such as in-network caching [9–17], mobility support [18–26], security and privacy problem [27–29] and more [30–32].

1.1 Background of Content-Centric Network (CCN)

CCN has three main data structures: the Forwarding Information Base(FIB), Content Store (CS) and Pending Interest Table (PIT). There are two types of CCN packet, *Interest* and *Data*. These packets carry the name of the content instead of the addresses of source and destination. Fig. 1.1 shows the entire packet forwarding process. At the beginning, requester will initiate the *Interest* packet and sent it to CCN node (e.g., router). When the CCN node receives *Interest* packet, it first attempts to satisfy it with contents by checking the Content Store (CS). If suitable content is found (Interest Hit), CCN node will copy the content and send it back to requester using a *Data* packet. Otherwise (Interest Miss), CCN node will forward the *Interest* packet via its outgoing interfaces by look up the Forward Interest Base (FIB).

FIB, similar to IP routing table, is used to find the outgoing interfaces that are connected to potential source(s) of matching *Data*. FIB may contains not only the physical interfaces but also the routes to the applications running on the local machine. Different with IP routing table which only allow one outgoing face, CCN FIB supports a list of faces. This means that CCN supports multiple sources with parallel routes to the *Data*.

After the *Interest* packets are forwarded upstream toward content source(s), they will be

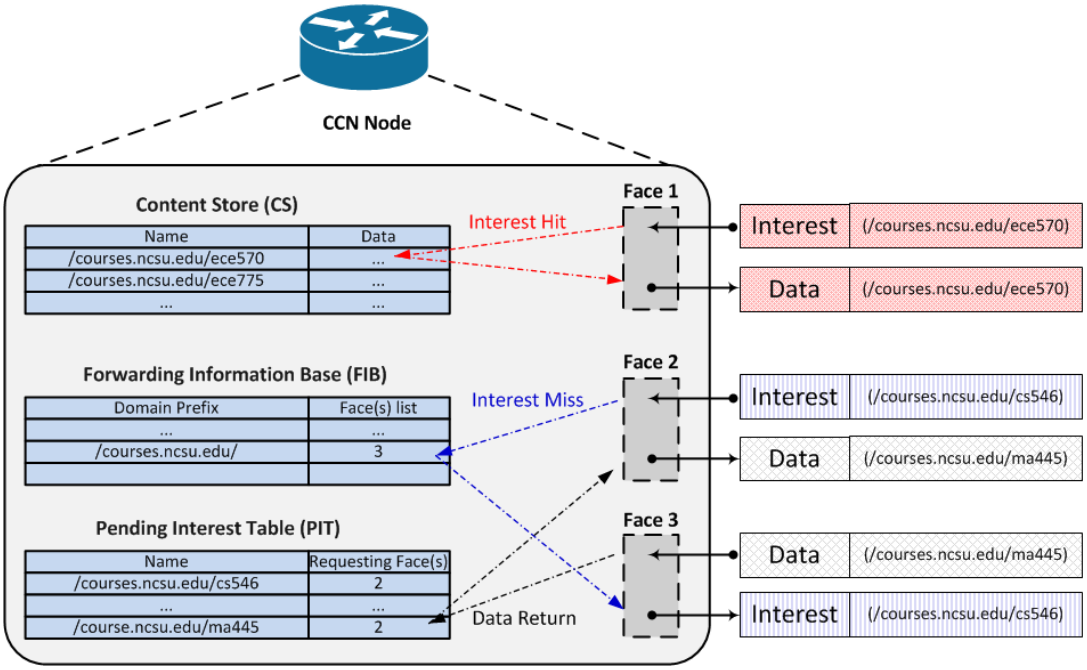


Figure 1.1: Forwarding Process at CCN node.

added into Pending Interest Table (PIT), which is used to route the returned *Data* packets to downstream to the requester(s). PIT can avoid the identical *Interest* packets to be forwarded multiple times. This *Interests* aggregation is one of the CCN features that improves the efficiency and avoid loop in CCN. When the *Data* packets arrive (Data Return), PIT will be used to route these *Data* packets back to requester(s). And CS will cache these *Data* packets if they are valid (fresh and non-duplicated).

The main difference between CCN and IP networks is that CCN is a receiver-driven model in which receiver uses the name of the content rather than the location for request. CCN deliver packets based on the name of the content in hop by hop manner, which is contrast to TCP/IP's connection-oriented end to end control. In another word, nodes in CCN are more interested in *what* contents you want instead of *where* are the contents. And this concept of CCN perfectly matches the user's goal of finding the content itself, while IP network maps this goal with addresses of content in a more complicated way.

1.2 Motivation and Open Questions

Among all these promising properties in CCN, in-network caching and mobility support play the most important roles in content delivery improvement. By adopting in-network caching, CCN

reduces the traffic redundancy since the contents that used to be stored at servers in IP network can be accessed at edge routers that are much closer to end users. In contrast to end-to-end location-dependent control in IP network, CCN is operated in connectionless manner, allowing users to re-sent the *Interest* packets for contents without complex connection establishment process.

1.2.1 Caching Efficiency Problem

Over the last few decades, caching has been extensively studied in World-Wide-Web (web caching) [33–35], Delay Tolerant Networks (DTNs) [36–38], Content Distribution Networks (CDNs) [39–42] and other networks [43–47]. Replacement policies obtain most of the attentions in caching study [48]. Several light-weighted replacement policies such as Least Recently Use (LRU), Least Frequently Used (LFU) are widely implemented in existing web caching and P2P network because of their effectiveness on keeping the important (popular or fresh) contents in the cache. As far as decision policies, most of the approaches are considered in some forms of collaborative or structured fashion. For example, [36, 38, 39, 44] proposed their efficient caching scheme in the cooperative manner so that cached data are shared and coordinated among multiple nodes; [49, 50] deployed the caching system with hierarchical structure to optimize tradeoff between data accessibility and accessing delay. However, these decision policies cannot satisfy line speed requirement in CCN simply because they require complex computation for information exchanges executed by multiple collaborative entities. Besides, the CCN architecture where named content caching and distributions are performed by every node makes caching in CCN a more challenged issue.

Although universal caching ($CacheAll$) decision policy and Least Recent Used (LRU) replacement policy are defined in CCN [4], some authors have already argued that they are impractical to be adopted in large network scenario [51, 52], with the reason that the relative small cache size (compared with total number of contents can be cached) will introduce high cache replacement error. Therefore, some researches have been conducted on CCN in-network caching with respect to cache decision policies (CDP) (e.g.,$ProbCache$ [12], $Betw$ [11]) and cache replacement policies (CRP) (e.g., LRU [9, 11, 29], MRU,MFU [53], $UNIF$ [17], $LRFU$ [13], Least Benefit (LB) [14]). For instance, $ProbCache$ [12] approximates the caching capability of a path and cache value contents probabilistically; $Betw$ [11] improves the caching gain by exploiting the betweeness centrality of network topology and cache content in a subset of nodes; $LRFU$ [13] assigns a *Combined Recency* and *Frequency* value to each object aiming to characterizing the likelihood that the object will be accessed in the future.

These studies greatly promote our understanding of caching performance in CCN and also propose innovative designs that further improve the content delivery efficiency in CCN. However,

most of these studies devoted their effort in the CRP rather than CDP. We argue the CDP should obtain more attention with three reasons. Firstly, CDP can avoid frequent cache eviction: due to the cache size restriction, the decision of caching one content means that one content will be evicted. A proper CDP can keep the Content Store more stable and robust as only important contents are cached. Secondly, CDP can minimize the control overhead from frequent FIB update among the network. With stable cache, less control messages are required to announce the cached content change. Thirdly, a well tuned CDP can prevent the content pollution caused by numerous unpopular contents requests [29].

Cache decision policies such as *ProbCache* [12] and *Betw* [11] are proposed for improving the caching efficiency. Both of their performance heavily rely on the network topology or available cache size. However, these topology-assistant or cache size based decision policies cannot fully utilize the cache capability in the network because they require network topology examination in advance. In addition, the work in [15] points out that the impact of topology on caching is limited and concludes that catalog and content popularity play the most crucial role in CCN. Since CCN has shifted the network paradigm from location to content, caching scheme should focus on content as well. And as proved in [54, 55], Internet traffic follow Zipf-like distribution, indicating that content popularity is the most important factor in cache policy design. Clearly, no agreement for CCN caching efficiency has been reached yet. Intuitively, if a cache policy can identify the content popularity and cache those most popular ones, the CCN caching performance will be significantly improved. So far, no such popularity-oriented decision policy has been designed to improve the caching efficiency of CCN, leaving it as an huge unexplored area. Thus, there is an open question:

Question: *Can we design a cache policy to optimize the caching efficiency in CCN?*

1.2.2 Handover Latency Problem

Popularity-oriented caching policies are effective for on-demand contents accessed by large number of user. However, they may suffer serious problem when we take the reactive mobility support in CCN into consideration. As shown in Fig. 5.1, mobile user will re-send the *Interests* that have not been received during the handoff (move away from one attachment point and connect to a new one). The delay of handoff could still be tolerable for non real-time applications such as email, message and web page browsing, but it can be significant and introduces serious degradation on the user-perceived quality of experience on delay-sensitive and real-time applications such as multimedia streaming, stock trading and online gaming. And the delay will be further exacerbated by popularity-oriented caching scheme: real-time contents will not be cached since

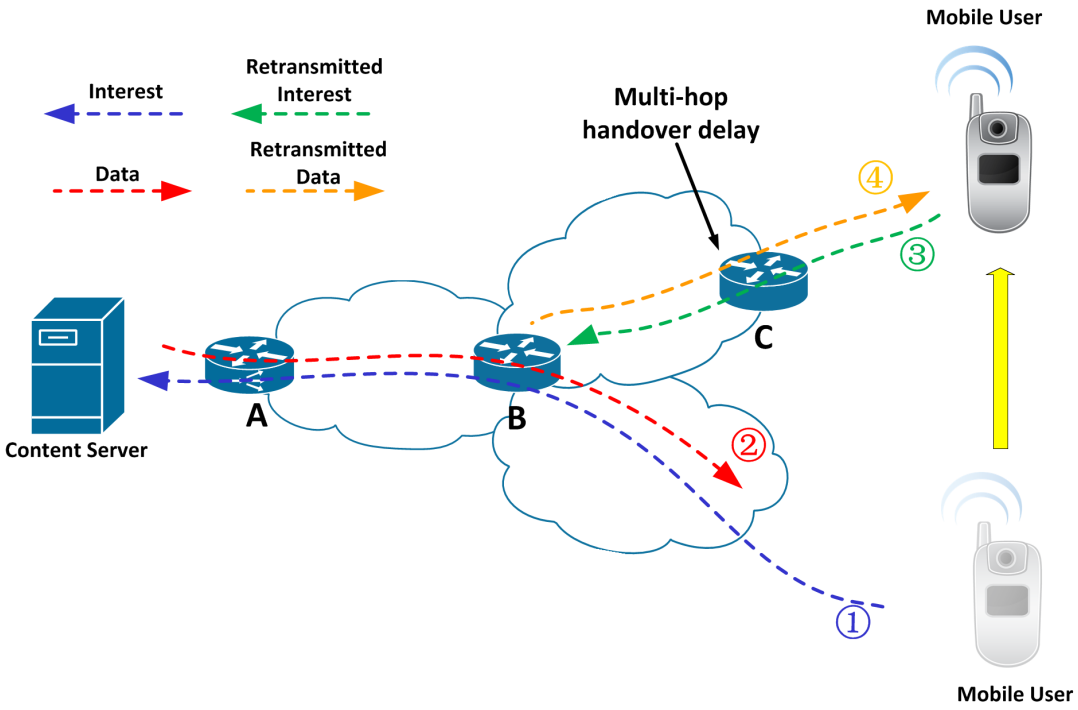


Figure 1.2: Mobility problem for real-time application in CCN.

most of them are unique (unpopular) and only be requested for individual users. We can expect that, by using popularity-based caching in mobile CCN scenario, the delay of real-time applications will be maximized and it is unable to satisfy the QoS requirement. To minimize the handover latency, we should let that retransmitted Interest packet to reach the content directly at newly connected router after handover. We define this one-hop-delay scenario as seamless handover.

Some studies have started working on reducing handover latency in CCN. For instance, [19] uses cost function to select appropriate neighbors and pre-cache the content. However it didn't provides clear details describing how contents are forwarded to these selected neighbors. Work in [18] proposes the mobility support schemes for CCN by adding the control panel in architectures of both mobile user and CCN node. However, there is no clear simulation results show how these control overheads and additional processing delay affect the performance. And since these schemes are at high level, leaving many designs and implementations issues unanswered. For instance, the architecture modifications on both mobile nodes and CCN nodes, makes it not backward compatible with current devices. Another primitive prototype application VoCCN [30] has been design on CCN to enable real-time voice service. However, it is a mapping of VoIP in IP network on CCN that cannot be applied to handle other generic traffic types.

Also, related works such as [20–23] are diving in the analysis and design of handover support for real-time applications in CCN or ICN architecture. However, they are either cannot achieve seamless handover or they require extra topology examination in advance. Thus, a fundamental question remains unsolved:

Question: *How to design a cache policy to support seamless handover proactively without topology examinations?*

1.3 Research Problem and Contributions

With the strong motivations and open questions identified in Sec. 1.2, we summarize our research problem in this work:

Research Problem: *Can we design a CCN cache policy to (1) improve caching efficiency and (2) support seamless handover ?*

Undoubtedly, this killing two birds with one stone design will bring huge benefits to CCN. First of all, the performance of popular contents dissemination, as the basic objective of CCN, is improved since the caching efficiency is optimized. Secondly, the mobility support capability of CCN is enhanced as the access delay for both on-demand and real-time contents are minimized. In addition, using in-network caching property for handover support can keep the CCN simple and more scalable because it is consistent with the concept of CCN that replacing *where* with *what*. Most importantly, this design is required for CCN to accommodate future mobile traffic demand. According to the statistics report from [56], the global mobile traffic, which was measly 1% and 4% at 2010 and 2011 respectively, hit 13% in November 2012. Cisco also released the Global Mobile Data Traffic Forecast report [57], and it predicts that mobile data traffic will grow at a compound annual growth rate (CAGR) of 66 percent from 2012 to 2017, reaching 11.2 exabytes per month by 2017.

As our goal in this research is to design a CCN caching policy to improve the caching efficiency and solve the handover latency problem, we take the following approaches:

1. **Extensive simulation studies on in-network caching efficiency in CCN**
   As existing efforts tend to optimize caching in different perspectives, we want to identify the fundamental factor or aspect that can significantly affect the caching efficiency in CCN. Therefore, we develop a scalable mobile CCN caching simulator $MCsim$ for the in-network caching simulation study. We thoroughly discuss the network and system settings

with a survey and define the caching simulation scenarios in our work.

2. **Efficient cache policy design to solve the caching efficiency problem**
   Study result shows that content popularity plays the most critical in caching efficiency. In addition, it explains the problems of existing caching policies and shed the light on our *CachePop*, a popularity-oriented cache policy design. In particular, we add a Request Table in CCN architecture for recording the arrival requests information. We specify the design details of *CachePop* algorithm and Request Table control method and validate our *CachePop* cache policy through *MCsim*.

3. **Proposed cache policy optimization to solve handover latency problem**
   In order to achieve seamless mobility support in CCN, we optimize *CachePop* with the neighbor graph generation and design the proactive caching routing mechanisms in mobile CCN scenario and propose *NG-CachePop* policy. In particular, we define two types of contents: on-demand and real-time, and apply different decision policies accordingly in *NG-CachePop*. We evaluate our *NG-CachePop* through *MCsim* in various mobile CCN scenarios.

Our contributions are summarized as follows.

- We develop a scalable Mobile CCN Caching Simulator (*MCsim*) which supports multi-parameters configurability in the mobile Content-Centric Network (CCN) scenarios.
- We conduct a thorough simulations study in CCN and the simulation results provide us a quantitative results for understanding the in-network caching performance in CCNs.
- We design the *CachePop*, a novel popularity-oriented decision policy in CCNs. *CachePop* differentiates the content popularity and cache non-duplicated contents along the delivery path to avoid caching pollution and therefore improve caching efficiency.
- We optimize the *CachePop* design with neighbor graph and proactive caching routing mechanism and propose Neighbor Graph Based *CachePop* (*NG-CachePop*) for achieving seamless handover in mobile CCN scenario.

1.4 Outline and Organization

The rest of this thesis is organized as follows. Chapter 2 introduces the existing researches on in-network caching and mobility management in various network architectures. In particular, we review several proposed designs and approaches for CCN by analysing their shortcomings in addressing the caching efficiency and handover latency problems. In Chapter 3, we conduct a

thorough simulation study via our *MCsim* with multiple network ingredients in both static and mobile CCN scenarios, aiming to identify the primary factor that impacts in-network caching efficiency in CCN. Chapter 4 presents the design details and simulation results of the popularity-oriented cache policy: *CachePop*. In Chapter 5, we optimize *CachePop* with proactively caching feature and propose the neighbor graph based caching policy: *NG-CachePop*. We validate our design in *MCsim* platform with convincing results. Finally, Chapter 6 concludes what we have achieved in this thesis and discusses the future works.

Chapter 2

Related Works

In-network caching is not a new concept as it has already been studied and implemented in many networks. Moreover, the handover latency problems in wireless networks have been discussed and a large amount of designs are proposed to address these issues. In this chapter, we provide a detailed review on the caching relevant researches with respect to caching architectures, cache decision and replacement policy among different networks in Sec. 2.1. In addition, we analyze the handover latency problems in traditional IP network and CCN scenarios in Sec. 2.2. Although the concepts of caching and handover in CCN and other networks are similar, due to the fundamental architecture change in CCN, most of the existent works cannot be directly implemented to CCN, making caching and handover latency in CCN a huge unexplored area.

2.1 Caching Schemes in Multiple Networks

Many studies have been done on cache policies in various network areas(e.g., Web caching [33, 35], CDN [39–42], DTN [36–38] and Ad-hoc network [43–45]). As a new network architecture, CCN also obtains lots of attention. Many existing caching replacement policies have been fully evaluated in CCN but much less studies work on decision policy. In this section, we first introduce the cache policy in World Wide Web(WWW) and Content Distribution Netork (CDN)area. Later on, we introduce the latest caching studies for CCN.

2.1.1 Caching Architectures

The performance of caching network dependents on the size of cache and the user community. The larger cache or user community size, the higher probability the contents that stored in the cache can be accessed again by other users. Therefore, caching system rapidly extended from a local cache within a single node(router, browser) to a shared cache system assisting users in certain area. Such lager scale caching system requires the caches to be organized and work

cooperatively to increase the cache gain. According to [58,59], caching architectures are mainly divided into two types: hierarchical caching architecture and distributed caching architecture.

Hierarchical Caching Architecture

Hierarchical caching architecture coordinates caches located in the same system to establish a caching hierarchy with institutional, regional, and national caches [60]. With hierarchical caching, client caches are located at the bottom level of the hierarchy. When a request is not satisfied by client cache, it is forwarded to institutional level caches. If the content is still not found, the request will be further redirected to higher level (regional and national). If request is not satisfied in the caching system, national level cache will forward the request directly to the origin content server. Once the content is found, it is forwarded down the hierarchy and leaves a copy in the intermediate caches. Therefore, latter request for the same content can be satisfied at intermediate caches rather than travelling to the original server.

Hierarchical caching architecture was first proposed in the Harvest project [60] and it is widely adopted in web caching researches (e.g., [49, 50, 61]). In [49], the cache is viewed conceptually as a low-pass filter with a cutoff frequency upper bounded with characteristic time and researchers proposed a cooperative hierarchical caching architecture which better improve the caching performance. [61] proposed adaptive web caching which equipped with the URL routing table and neighbor cache contents to forward all missing queries quickly and efficiently. With hierarchical caching structure, popular contents can be efficiently disseminated in the system, providing the advantages such as high efficient bandwidth utilization and low content access delay. However, hierarchical caching may introduce other problems: (i)the higher level caches could become bottlenecks, as all the missing requests are forwarded to high level caches, leading to longer queuing delays (ii) same copies of the content are stored redundantly at different level of caches and (iii) significant coordination effort for placing the participating caches to set up the caching hierarchy.

Distributed Caching Architecture

Contrary to the hierarchical architecture, distributed web caching allocates the caches at the edge of the network without intermediate caches. Low level caches work in cooperative manner and if the request is not satisfied at one cache, it will be redirected to another edge cache based on the meta-data information about the content of every other cache. Compared with hierarchical caching, distributed caching allows most of the traffic remains at low network level, which has much large amount of contents and less traffic congestion. In addition, distributed caching allows better load sharing and are more fault tolerant.

Distributed caching architectures are also implemented in Content Distribution Network

(CDN). CDN [40,62,63] are designed to improve performance and scalability of content dissemination in WWW by replicating content from the original server to distributed nodes, so-called surrogates. In order to keep the correct mete-data for request redirection, different techniques are proposed such as query-based approach [64], digest-based approach [65, 66] and directory-based approach [67]. Query-based approach [64] allows the cache to broadcast a query after the cache miss. Digest-based approach [65,66] avoids the inefficient queries process by containing a digest or summary of content held by other cooperating caches. Directory-based appraoch [67] implements a centralized server to keep the content information of all cooperating caches. Actually, many other networks utilize distributed caching architectures as well. For example, [36–38] proposed cooperative caching designs in Delay Tolerant Network (DTN) based on popularity and social community. Similar to DTN, several works [43–45] introduce caching schemes for ad hoc network in cooperative manner. Because lower level nodes are the majority in these two networks, distributed caching architecture is particularly suitable as it doesn't require intermediate caches. Although the concept of content replication in CDN is similar with in-network caching in CCN, the operation complexity introduced by the cooperative manner makes most of the existing caching approaches inapplicable to CCN. In addition, existing caching schemes for CDN highly rely on the location-dependent redirection, which is in conflict with the fundamental design architecture of CCN.

2.1.2 Cache Replacement Policy

Caching replacement refers to the process that takes place when the cache becomes full and old content must be removed to make space for new ones. For most the cache replacement policy (CRP) studies, the main goal is to cache more important contents and evict the less important ones. Importance mostly refers to the number of request, request interval or cost. CRP can be classified into the following three categories as suggested in [48].

Recency-Based Strategies

Least Recently Used (LRU)is the most widely adopted CRP for caching system. LRU evicts the content which was requested the least recently. Many caching schemes and strategies are proposed for various network and most of them are more or less the extensions of LRU. For example, LRU-Threshold [68]is the same as LRU but content size larger than a pre-defined threshold size is not cached. SIZE [69] evicts the largest content, Otherwise LRU is applied to contents that with the same size. LRU has been applied successfully in many different areas. And now it is commonly used in CCN studies [11, 12, 15, 29, 70] and CCN proposal [4] because of its simplicity in allocating cache for more popular contents based on the request pattern, thus increasing the cache hit ratio. Some CCN studies used or proposed more intelligent and

adaptive schemes such as Most Recently Use (MRU) [53] and Least Recently/Frequently Used (LRFU) [13]. LRFU considers the tradeoff between recency and frequency and it performs differently based on the topology location of the cache.

Frequency-Based Strategies

Frequency-based strategies use frequency as the criteria for content replacement. Least Frequently Used (LFU)is another traditional replacement policy and most of the frequency-based schemes are based on it. The concept of frequency-based strategies consider the fact the different contents with different popularity values which result in different frequency values. By tracking these frequency values, LFU can determine which content can be evicted. Prefect LFU counts all requests to every content in the past for completeness, but suffer severe overhead problem. In-Cache LFU decreases the overhead by counting the contents that stored in the cache only. Due to the space constriction, most of the frequency-based schemes are based on In-Cache LFU. LFU-Aging and LFU-DA [71] improve the LFU by adding the aging effect to avoid outdate contents remain in the cache. swLFU [72] applies weighted frequency counter provided by the server to influence normal LFU schemes. In CCN, [53] uses Most Frequently Used (MFU) in the network to avoid duplicated contents in the system.

Key-Based Strategies

Besides recency and frequency factors, many other key factors are used and evaluated in caching studies. Based upon these factors, replacement policies determine which content should be evicted. For instant, Hyper-G [69] combines LRU, LFU and SIZE and breaks ties by using the recency of last use and size. UVA [73] analyzes the user visiting action to keep contents that clients are more likely to access in the future. [74] calculates the individual cache profit of certain versions of a video object by considering the popularity, trascoding and average access duration, and removes less profits content. In the context to CCN studies, LB [14] takes into account the distance factor and forwarding process when replacing the contents. Work in [53] proposes a ICN architecture named MultiCache which based on two primitives: multicast and caching. Age-based caching [75] use the content popularity and location to guide the age factor for each content. The content is removed from the cache when the age expires.

Randomized Strategies

Randomized strategies randomly find a content and remove it for new one. [35] selects a set of contents randomly and evicts the least useful one in the sample. The usefulness of a content can be determined by any utility function, making this approach more adaptive. In [15], authors consider that CRP in CCN must happen at line-speed and therefore they evaluate Uniform

Random Replacement (UNIF), First in First out (FIFO) and BIAS(randomly select two objects and most popular one is replaced) in their work. Surprisingly, the results show that the performance difference across replacement policies is minimal.

2.1.3 Cache Decision Policy

Compared with extensive cache replacement studies in different networks, cache decision policy has not been well studied yet. Most of the caching designs simply cache every arrival content and lets cache replacement policy to determine whether the content should stay in the cache or not. Meanwhile, many caching systems work cooperatively in order to increase the content diversity in the subnetwork. [76] evaluates the ad-hoc network caching and introduces Distributed Greedy Algorithm (GDA) in which cache decision is based on benefit of the content. The benefit is computed by taking local traffic, access frequency into account and most beneficial content is cached. QIRMA [77] for CDN classifies contents into two classes and caches most frequently accessed contents that from class I in cluster with high weight value, while caches least frequently accessed contents that from class II in cluster with low weight value. Weight vector includes available capacity , CPU speed and access latency. QIRMA improves the system scalability by distributing the load across multiple caches and avoid the hot cache becomes a bottleneck. Other works such as DSR [78]and DAWCD [79] replicate demanded pattern content and popular content respectively so that to reduce the bandwidth and access cost.

Among the caching studies in CCN, much attention has been devoted to cache replacement aspect. There are still a lot of unexplored areas in cache decision policies and therefore more research efforts are required. An efficient CDP can significant improve the cache robustness as it filtrates contents when they arrive and avoid valued contents to be evicted (replacement error), which is likely happen in relative smaller cache size case. Because of the line speed requirement in CCN, most of existing caching decision works cannot be applied directly to CCN as they require high computation complexity. Hence, current works for cache decision policy in CCN are relatively simply and they can be mainly divided into three categories: universal caching, random caching and topology-assistant caching.

Universal Caching

The general assumption in decision policy is that any new contents should be cached. And this universal caching is used in CCN proposal [4]. Universal caching is perfect in design perspective because it is simple and can be applied to all content from all users in all CCN nodes. However, some works already argued that such indiscriminate universal caching strategy is costly and sub-optimal, indicating that it is hard to be implemented in large CCN [11, 51]. Recalling that Internet requests follow Zips'like distribution [54, 55] , we know there is a reasonable sized set of

popular contents, and a very long tail of contents that are requested by small population. And today's use of CDN has showed that moderately sized edge caches are sufficient to handle the popular contents. Considering the very long tail of contents in the Internet, we can expect that the effectiveness of universal caching barely increases with the cache size which is extremely smaller compared with contents that can be cached.

Random Caching

In [17], authors used random fractional caching, where each packet has a pre-set probability of getting indexed and cached. Similar ideas are used in [46] and [80]. Actually, universal caching is a extreme case of this approach as the probability $P \in (0, 1]$ in random caching is fixed to 1 in universal case. [17] uses fixed probability $P \in 0.75, 09$ in their simulation process. However, [15] uses the same probability value in a thorough simulation and shows that there is no significant performance improvement compared with university policy and LCD. The reason is obvious and easy to understand as all these cache policies treat every content individually without considering their interconnecting properties in a time interval.

Topology assistant Caching

Realizing that caching indiscriminately does not necessarily guarantee the highest cache hit ratio, some researches have moved onto network topology assistant caching design. A recent work in [11] proposes a caching scheme that allow contents to be cached only at "important" nodes, which is determined by the betweenness centrality of nodes that lie on the delivery path in a network. This caching approach can keep the contents to be cached at a node where a cache hit is most probable to happen and meanwhile reduce the cache replacement rate. The author claim that their caching schemes can be applied to any topology since the caching decision is solely based on the node itself. And the simulation results show that by using this scheme, the Interest packets can be satisfied with less hop comparing with CacheAll decision policy. However, we doubt that this caching scheme cannot satisfy the mainstream traffic pattern for three reason. First of all, the design idea behind these approaches is still location-oriented, from which CCN architecture shifted. One potential problem is that more nodes in network will prefer to route their Interest packet to these "important" nodes since they provide higher hit ratio, leading to significant traffic load and congestion on this area. Secondly, based on the algorithm of this topology assistant caching, nodes which have higher betweeness centrality will located at upper level of the network topology, leading to more contents to be cached far away from users. This result is opposite to our current understanding that putting the contents close to users [citation of CDN]. Thirdly, such scheme will decrease the content diversity among the entire network as less content can be cached.

Another similar design is introduced in [12]. The authors propose ProbCache to fairly allocate cache space for multiplex contents of different flows in caches along a shared path, by probabilistically approximating caching capability of a path and caches contents. Based on this scheme, node which receives the content will take remaining caches and hops into account when making cache decision. In contrary to [11], [12] makes node that located closer to the end user have higher probability to cache the content. However, this scheme requires additional network information such as cache size on the path and link speed which are hard to estimate in larger scale network topology as they are not stable all the time. While the common problem of these two schemes is that both of them only consider *where* to cache rather than *what* to cache. Actually, the latter one is more fundamental problem needs to be considered since small amount of popular contents attract the majority of the requests.

2.2 Mobility Management in Wireless Networks

With the increasing demand of large volume data and real-time services, wireless networks should be able to support the traffic consistency with different Quality of Service (QoS) guarantees. In IP based wireless networks, Mobile IP (MIPv4) [81] is the most well-known mobility schemes that improves the node mobility by redirecting the data packets to its current location through a tunnel. Besides MIPv4, there are many other works on mobility and handover management such as regional registration [82], SIP [83] and S-MIP [84]. However, due to the IP address blinding, these approaches suffer from problems such as handover delay, control overhead and triangular routing when handling the content delivery between initial point of attachment (PoA) and current connected point. CCN architecture splits the address with identity, as all entities are uniquely named so that contents are routed by names rather than addresses. Because of these fundamental architecture change, many studies [18, 19, 21, 22, 25] work on mobility and handover support in CCN.

2.2.1 Mobility and Handover Support in IP networks

MIPv4 [81] defines a home network where Mobile Node(MN) is initially connected and assigned a permanent IP address that used to identify the MN; it also defines foreign networks that MN visits. Two entities namely Home Agent (HA) and Foreign Agent (FA) are located in these two networks respectively. When MN moves to foreign networks, it will be assigned a Care of Address (CoA) through DHCP. And HA and FA are responsible to forward the data generated by Correspondent Node (CN) that routed based on home IP address to CoA. Although MIPv4 solves the IP address registration problem by introducing HA and FA, the registration delay between these two entities can be very long if the distance between visited network and home

network is large. Mobile IP Regional Registration (MIP-RR) [82] attempts to minimize the signaling delay by performing registration locally in a regional network. MIP-RR introduces a new entity called Gateway Foreign Agent (GFA) which is similar with FA. The address of GFA is advertised by the FAs that are located in the same visit domain. If MN is moving between different FAs within this visit domain, it only needs to make a regional registration to GFA instead of HA. For the data forwarding, HA first intercepts the packet and forwards it to registered GFA. And GFA will further redirect the packet to corresponding FA that located in its domain. MIP-RR reduces the frequent registration overhead and signaling delay by putting the registration locally.

SIP [83] is an application layer multimedia signaling protocol that developed by IETF. MN uses a INVITE message to initiate a session with the CN and it sends RE-INVITE message to maintain the same Call-ID when MN accesses to a new network and obtains a new IP address. After that, location information stored in home SIP server can be updated. In order to further reduce the delay during the handoff process, [84] developed a seamless handoff architecture for Mobile IP called S-MIP. In S-MIP, the network uses location and movement patterns of MN to 'instruct' the MN when and how handoff should be carried out. By using signal strength and storing the history of the locations, Decision Engine (DE) is able to obtain the movement pattern of associated MNs and the information of access routers. Therefore, DE can make the 'smart' handoff decision for the MNs to reduce the handoff delay and improve the QoS.

2.2.2 Mobility and Handover Support in Content-Centric Network

Unlike IP network, CCN doesn't worry about the reestablishment between two end users since CCN is connectionless. So most researchers are mainly focus on the connection consistency during the movement. Even though mobility support is the inherent benefit that come with CCN, it cannot satisfy the delay requirements of real-time contents if pure popularity-oriented replacement policies are used. Topology-assistant decision policies that allowing more contents to be cached at core rather than edge of the network seem to perform better because users have higher probability to re-access the contents. However, they will suffer the side effects as the congestion and processing delay at these "important" nodes can counter such advantage or make it even worse. To this end, the problem of seamless handover in CCN still remains unsolved. Several handover support designs for CCN or ICN architecture have been proposed to address this problem. These solutions can be classified into three types: reactive approaches, vertical handover approaches and proactive approaches based upon *where* requested contents are cached.

Reactive Approach

In reactive approach, mobile user will inform its point of attachment (PoA) before moving to another area. When the PoA receives this notification, it will cache the contents that requested by the mobile user after the disconnection. As long as mobile user reconnects with a new PoA, it provides the previous PoA identity to current PoA so that current PoA can obtain the contents cached during user's disconnection period without re-sending the request to the network. One of reactive approaches is proposed in [22], in which nodes use *Request()* and *Respond()* messages to disconnect the old broker and forward cached contents to new broker respectively. The brokers use the Subscription Table to handle these connection activities. Particularly, [25] handles the source mobility problem by redirecting incoming *Interest* packets via the tunneling between previous PoA and the new one. This redirection avoids the *Interest* packets missing due to the location change of source nodes. Another similar work [21] uses the indirection point to record the location change and re-deliver contents to new PoA. These reactive approaches allow new PoA to get the contents from previous PoA directly rather than resend the Interest packet to the content servers. However, they have the disadvantage as the delay will be increased in communication period between two PoAs. The mobile user cannot receive contents immediately because the contents have not arrived at current PoA yet. Obviously, reactive approaches still suffer multi-hop handover delay and therefore cannot achieve seamless handover requirement to avoid significant QoS degradation for real-time contents.

Vertical Handover Approach

Similar to soft handover in cellular network, vertical handover approach establishes the connection with new PoA before the connection to the old PoA is broken. This approach can better maintain the communication consistency since mobile user is receiving contents during its movement. One example of this design is [24]. The authors design a content delivery scheme that requires CCN nodes to use multiple interfaces for communications in overlap area of PoAs. And a Flow Mapping Agent (FMA) is proposed to manage the connection information. Authors argue that this approach can enhance content delivery efficiency as several interfaces are receiving contents simultaneously. However, this approach is hard to be achieved as it requires CCN nodes to carry multiple interfaces for the same network. In addition, the author only evaluate their design performance in a simple line shape network topology, in which two PoAs are directly connected. This design may has problem when two PoAs are not in the same subnet area and therefore it cannot solve the multi-hop handover latency problem in CCN.

Proactive Approach

Compared with reactive and vertical handover approaches, proactive approaches are more feasible: contents are delivered to new PoA that mobile user will connect to in proactive manner. So mobile user can receive the contents that requested in the previous PoA immediately after the handover. Many researchers have devoted their effort on proactive approaches and proposed several handover support schemes [18, 19].

In our work, we focus more on the user mobility. [19] proposes Selective Neighbor Caching (SNC) to enhance the user mobility in ICN architecture. The key idea of SNC is to select an optimum subset of neighbor proxies to cache the mobile user's subscriptions (requested contents) via the target cost function. This approach reduces the delay for real-time contents delivery but it also requires many extra information (e.g., user mobility, delay, cache cost) in decision process. Another work in [18] proposes three schemes to deal with the mobility problem. This work fully explore the potential network properties of CCN and provides a comprehensive view for addressing handover support problem. However, among these three schemes, both point of attachment and Rendezvous point based approaches involve additional control panels and complicated naming systems in packet routing process. While the Multicast based approach that using multi-path interest forwarding feature in CCN is not cost-efficient. It may result in significant congestion problem when large number of mobile users are exist. More importantly, no work so far has ever considered how their handover support scheme are compatible with in-network cache policy.

2.3 Summary

In Fig. 2.1 and Table 2.1, we summarize aforementioned related works in CCN with respect to cache policies and mobility support. From the discussion above we can know that most of current researches focus on either the in-network caching performance (especially on CRP) or mobility support. No work so far has ever considered how the mobility support designs are compatible with in-network caching scheme. And none the them has ever considered to solve handover latency problem through the cache policy aspect. Note that our work fills this void in CCN by taking these two issues into consideration.

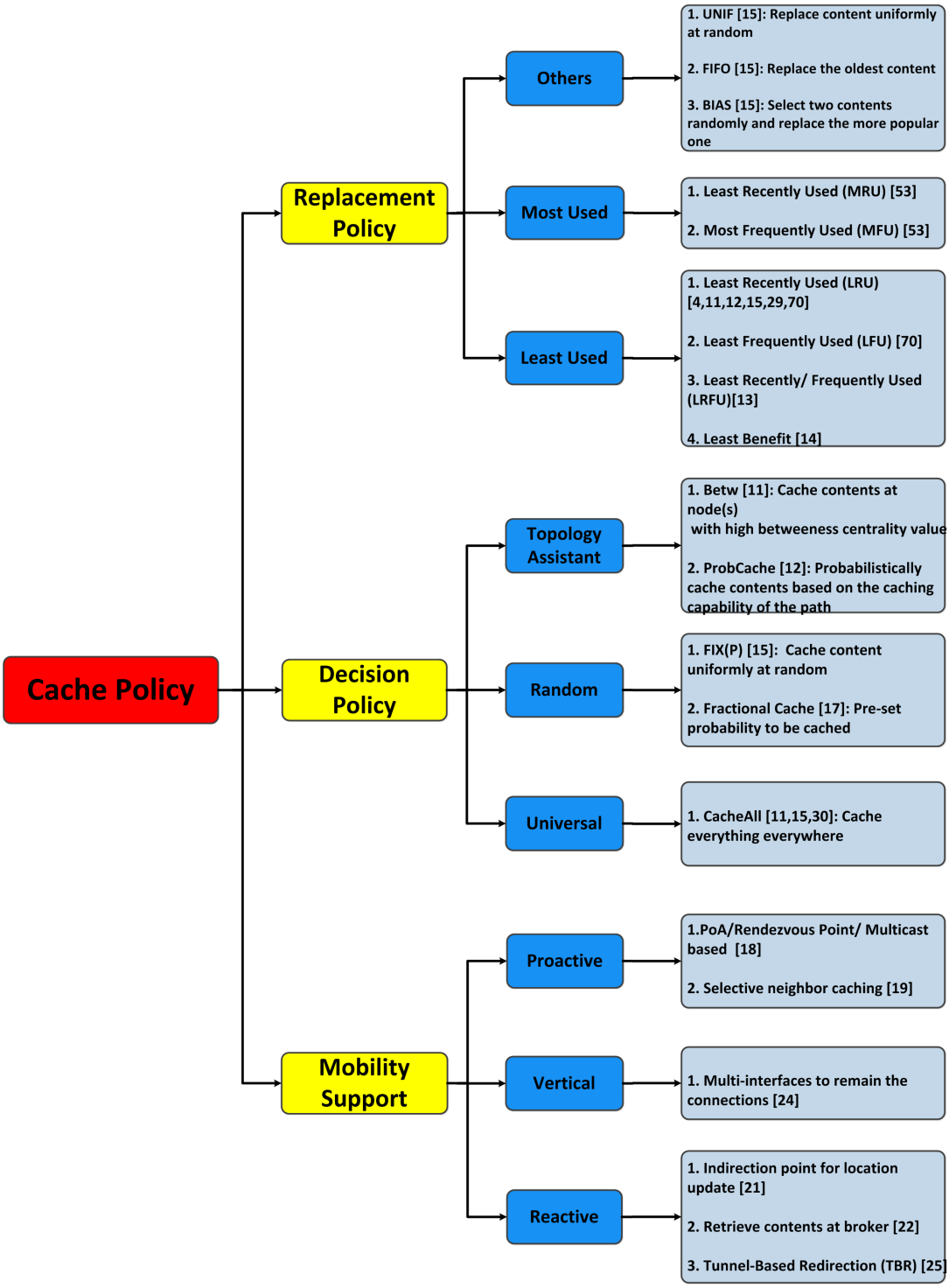


Figure 2.1: Related works in CCN.

Table 2.1: Comparison of Related Works on Caching Policy and Handover Support in CCN.

Cache Policies	CRP	Least Used	LRU [4, 11, 12, 15, 29, 70], LFU [70], LRFU [13], LB [14]
		Most Used	MRU, MFU [53]
		Other	UNIF, FIFO, BIAS [15]
	CDP	Universal	CacheAll [11, 15, 30]
		Random	FIX(P) [15, 17]
		Topology	Betw [11], ProbCache [12]
Mobility Support	Reactive		[21, 22, 25]
	Vertical Handoff		[24]
	Proactive		[18, 19]

Chapter 3

Understanding of Caching Performance in Content-Centric Network

Considering that our objective of this research is to design a cache policy to enhance the caching efficiency in Content-Centric Network (CCN), we believe that it is important to investigate and understand how in-network caching perform in CCN in advance. Therefore, in this chapter, we conduct extensive simulations and measurements for evaluating the affects of environment factors (e.g., α value in request traffic, catalog size, network topology) and system parameters (e.g., cache size, cache policy) on caching performance in CCN. We attempt to address several essential questions as the first step to comprehensively evaluate the performance of CCN: 1) Why existing policies are not efficient in CCN? 2) What is the primary factor that affects the caching efficiency in CCN? In order to answer these questions, we develop a simulator: Mobile CCN Caching Simulator ($MCsim$) that can be utilized in both static and mobile CCN scenarios with configurable parameter as aforementioned. Based on the simulation results, we identify the primary factor that impacts the cache efficiency and summarize four design principles that guide our future cache policy design in next chapter.

The organization of this chapter is as follows. Sec. 3.1 introduces the motivation of this simulation study on CCN caching performance. Sec. 3.2 describes the design details of Mobile CCN Caching Simulator ($MCsim$) development. In Sec. 3.3, we discuss and define the simulation scenarios in our caching study. Finally, we present and discuss the simulation results in Sec. 3.4.

3.1 Motivation and Related Works

As introduced in Chapter. 2, caching has already been extensively studied in a variety of network areas such as Web caching, Content Delivery Network, Delay Tolerant Network and Ad-hoc Network. However, the CCN architecture where named content caching and distribution are performed by every nodes make caching in CCN a huge unexplored area. The potential impacts of caching in CCN have not been thoroughly assessed yet. So far, there is no consistent or clear picture showing us how in-network caching performs in CCN and what is the primary factor(s) that affects the efficiency of content retrieval. These uncertainties greatly hinder our understanding on in-network caching performance in CCN.

The reason that lack of comprehensive caching performance evaluation in CCN is partly due to the large scale of Internet catalog sizes, to the complex router deployment and to the user behavior. For instance, in contrast with limiting the contents to be cached at some particular routers, CCN applies universal caching, through which each router can cache any contents so that to offload the traffic among the network. In addition, different with sufficient understanding on Web caching or in existing networks, the scientific community has not reached the consensus on the evaluation scenario for CCN, making caching performance evaluation in CCN a more challenging issue. In fact, a thorough study on CCN caching performance can be of great help to further optimize the cache policies and therefore enhance the overall CCN performance.

Some researches such as [11, 12, 14] have been done on the cache policy design in CCN with the goal to improve the efficiency of content caching. Another works [19, 21] aim to reduce the handover latency in CCN via either proactive or reactive caching algorithms. Among these researches, various simulations under different scenarios are conducted for performance evaluation. However, none of them provide an comprehensive evaluation results demonstrating the caching performance in CCN under multiple system or environment parameters. We list several evaluated parameters utilized in related works in Tab. 3.1. As we can see, most of the works evaluate the cache size factor since it is the most intuitive factors for enhancing the content performance. We also observe that only a few works evaluate the CCN caching performance by using different α value, content types or considering mobility. For instance, [12] simply evaluates the cache size effect on CCN caching performance in static scenario with binary tree topology and fixed α value equals 0.8. Work [15] presents a simulation study of CCN by considering several aspects including, α value of request traffic, cache catalog size, topology and cache policies but without content type and mobility scenario. We argue that such limited evaluations are not sufficient to analyze and target the primary factor that cache policy design should focus on. To this end, an comprehensive evaluation on CCN in-network caching is urgently needed before we start our cache policy design for improving caching efficiency and reducing handover latency in mobile CCN.

Table 3.1: Parameters of CCN caching evaluation in related works.

Related works	Evaluated parameters					
	α value	Content type	Cache size	Topology	Caching policy	Mobility
[10, 12, 13]	✗	✗	✓	✗	✓	✗
[11]	✗	✗	✗	✓	✓	✗
[14]	✓	✓	✓	✓	✓	✗
[15]	✓	✗	✓	✓	✓	✗
[19]	✗	✗	✗	✗	✗	✓
[21]	✗	✓	✗	✗	✗	✓
our work	✓	✓	✓	✓	✓	✓

In our work, we take all six parameters listed in Tab. 3.1 into consideration during the caching evaluation. Our goal is to conduct a very thorough simulation study of CCN caching performance aiming to provide a clearer view for understanding how those parameters impact in-network caching in CCN. Yet, no CCN simulator has been developed for caching evaluation with multiple parameters in mobile environment so far. For example, CCN simulator provided in [15] is good for caching evaluation but only in static scenario. Hence, we advocate the need to design a mobile CCN simulator for both the caching study and design performance evaluation.

3.2 Mobile CCN Caching Simulator ($MCsim$)

In this section, we describe the development of our customize CCN simulator. As we focus on the evaluation of caching performance and seamless handover support in CCN, our simulator should be able to fulfill these requirements. We develop Mobile CCN Caching Simulator ($MCsim$) based on Java platform and MySQL through JDBC. The key feature of $MCsim$ is the mobility support for CCN users who are sending requests (*Interest*) for contents during their movement. Also, $MCsim$ enables multiple configurable parameters including content popularity, request traffic type, cache size, topology and cache policy. *CacheAll* decision policy and Least Recently Used (LRU) replacement policy are used by default.

$MCsim$ has three modules and they are User module, Router module and Server module. All modules have Receiver (Rx), Transmitter (Tx) and Traffic Manager (TM). Rx forwards the packets to Traffic Manager (TM) when it receives packets from the network. Tx is responsible to send the packets received from TM to the network. Traffic Manager is the main component for processing data (*Interest, Content* and simulation results). In different modules, TM has particular functions and schedules. We will introduce these three modules with details below.

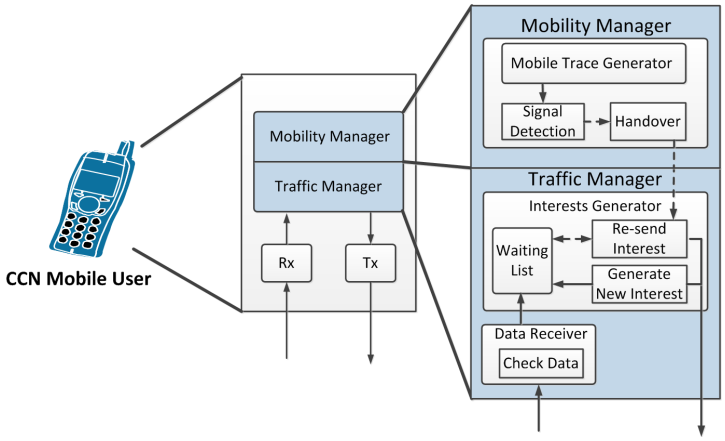


Figure 3.1: Structure and traffic operation in user module.

3.2.1 *MCsim* User Module

CCN user is located at the bottom level of the network for *Interest* generation and *Data* collection. In user's perspective, the delay of receiving the Data is the most important factor that concerned. Fig. 3.1 depicts the general architecture of user module in *MCsim*. In this user module, Traffic Manger (TM) determines how to handle the data, including *Interest* generation and received content checking. At the beginning, Interest Generator in TM generates new *Interest* packets following particular distributions. Through Tx, *Interest* is sent to the connected CCN router and the *Interest* ID is added into the Waiting List. Meanwhile, Rx will listen the incoming traffic; once the *Data* is routed back, Rx will collect and forward it to Content Receiver to check the user ID of received *Data*. If the user ID matches the node ID, Content Receiver will notify the Interest Generator to delete the corresponding Interest ID in the waiting list. In our simulation, we do not consider the packet loss as we are more interested in the caching performance.

In order to evaluate the caching performance in mobile CCN, we implement Mobility Manager (MM) at the top of Traffic Manager. MM generates mobile traces and detect the handover during the movement. In the handover process, MM disconnects the previous router and connects to a new router that located the closest to its current location. Also, Traffic Manager will handle the handover case by re-sending the *Interests* in the waiting list.

3.2.2 *MCsim* Router Module

CCN router plays the most vital role in CCN because it reduces the content delivery delay and network bandwidth usage by storing contents in the cache to satisfy the request traffic. Fig. 3.2 depicts the architecture of *MCsim* router module. Traffic Separator collects the data from Rx

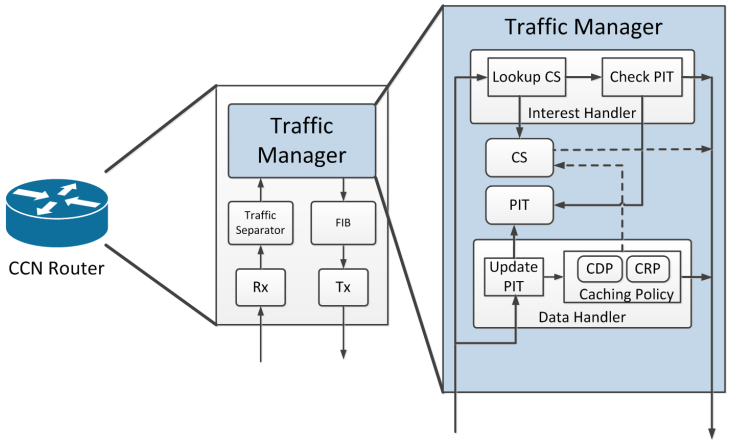


Figure 3.2: Structure and traffic operation in router module.

and check its type(*Interest* or *Data*). If received content is *Interest*, it is delivered to Interest Handler. Otherwise, it is delivered to Content Handler in Traffic Manager (TM).

When an *Interest* packet arrives at TM, TM will check the PIT first to see whether it has a duplicate *Interest* that been sent before. If yes, TM will update the PIT by adding the incoming interface to the existing Interest item. And the *Interest* will not be forwarded to the network for bandwidth saving. If no duplicate *Interest* is found in the PIT, Interest handler will look up the Content Store (CS) to see whether it has stored content that can satisfy this *Interest*. If content is found in CS, this content will be delivered back to user. Otherwise, TM will forward this *Interest* to another router heading to the content server. Also, TM adds this *Interest* into the PIT to indicate that future *Interest* for the same content should wait here. Before sending out the *Interest* packet, this router module uses FIB to determine which outgoing interface should be used.

When a *Data* packet arrives at TM, TM will update the PIT and deliver the *Data* to other routers or users based on the PIT information. Also, TM applies Cache Decision Policy to determine whether to cache arrived *Data* or not. If it decides to cache the *Data* but the cache is full, TM will use Cache Replacement Policy to remove old *Data* in cache. After this procedure, the *Data* will sent out.

Outgoing packets from TM will go through Forwarding Information Base (FIB) which applies routing protocol to guide the packets to correct interfaces. In this simulator, we assume routers know the route to all content servers. Without loss of generality, we use the shortest path as the content delivery path in this work. And the contents will be delivered to users based on the path of *Interest* in reversed direction.

Caching performance in routers is the most essential factor that affects the overall CCN performance. The more *Interest* can be satisfied at lower layer network, the more bandwidth

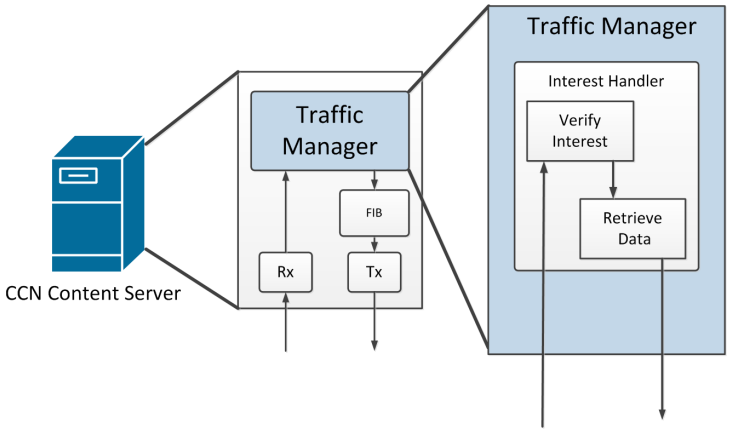


Figure 3.3: Structure and traffic operation in content server module.

saved. Hence, our main simulation results are collected at *MCsim* router module.

3.2.3 *MCsim* Content Server Module

In CCN, content server is the final node that *Interest* is forwarded to when it cannot be satisfied along the forwarding path in the network. Once the content server receives the *Interest*, it replies it with the corresponding *Data*. Fig. 3.3 demonstrates the architecture of *MCsim* content server. The major task for TM in this module is to generate contents so that to satisfy the received *Interest* and send the *Data* back to the routers based on FIB.

3.2.4 Simulation Results Collection

In *MCsim*, simulation result collection is embedded in different modules. In user module, the result collections between two versions are slightly different. In the general version user module, *MCsim* do not collect result data as we assume that the connections between users and routers are perfect for simplicity. While in the mobile version *MCsim* user module, *MCsim* collects the result about re-transmitted Interests especially for real-time traffic. Our goal is to demonstrate the hop distance ratio of contents whose Interest are re-transmitted after the handover. The main result collection effort is devoted onto router module, in which the in-network caching schemes are deployed.

Table 3.2 shows the details of collected results during the simulation among three modules in *MCsim*. Besides common cache hit and miss results, we also collect the contents diversity in the cache during the simulation. This is because content diversity is a more straightforward metric for us to analyze content distribution in the cache. Content diversity is classified into four categories. First category refers to the top popular contents with the rank matches to the

Table 3.2: *MCsim* result collection table.

Module	Collected result	Meaning
User	**OD_Interest**	number of retransmitted Interest for on-demand traffic
	RT_Interest	number of retransmitted Interest for real-time traffic
	OD_AccHops	hops of Interest packets to reach the on-demand contents
	RT_AccHops	hops of Interest packets to reach the real-time contents
Router	**Cache Hit**	number of arrival Interests that hit the requested contents
	Cache Miss	number of arrival Interests that miss the requested contents
	AccHops	accumulated hops of all arrival contents
	Diversity 1-4	realtime content popularity distribution in local cache
Server	**OD_Interest**	total number of arrival on-demand Interest packets
	RT_Interest	total number of arrival real-time Interest packets

current cache size c: $[1, c]$. Second and third categories refers to the same amount of contents rank from $[c+1, 2c]$ and $[2c+1, 3c]$ respectively. The last category refers to all the rest relatively unpopular contents. Accumulate Hops (AccHops) is obtained from each arrived packet which record the hops that it has travelled.

3.2.5 Header format of packet

The header of packet is shown in Table 3.3.

Table 3.3: Header of packet.

1. Type	2. ContentName	3. ServerID	4. UserID	5. Hops
1. Type: The first header field indicates the type of this packet. 0 represents Interest and 1 represents Data.				
2. Content Name: This field shows unique name for this content.				
3. Server ID: If no router caches the content whose name matches field two, the Interest will be routed to the content server with this server ID.				
4. User ID: This field is used by routers to forward the content back to user.				
5. Hops: This field is used for recording the one way hop count. It is 0 by default when Interest is generated. Every time the Interest is received by a node, Hops will increase by one. When this Interest is satisfied by the content or server, Hops value will be copied to content packet.				

3.3 System and Scenario Description

In this section, we discuss the potential parameters that might affect the caching performance in CCN. We conduct a survey on caching parameter settings in related works and classify the values of these parameters in Table 3.4. In addition, we summarize the system environment and parameters values defined in our simulation in Table 3.6.

Table 3.4: System parameters setting in related studies.

Parameters	Values
α	0.68 [12], 0.9 [29], 1.0 [9,11,13], 1.5 [15], 2 [85], 0.64-0.83 [55], 0.5-0.9 [14], 0.5-2.5 [16], 1-2.5 [86], 0.7-2.4 [70]
Chunk Size	4KB [70], 10KB [15, 86]
Content Size	1KB [87], 80KB,1.6MB [70], 6.9MB [86], 10MB [10, 15, 16], 1 Second Traffic [12], 1 [11, 13, 14, 29]
Cache Size	100KB-6.4MB [87], 40MB [70], 100MB [75], $10GB(10^3)$ [15, 16], 1GB-16GB [86], 100 [11], 500 [14], 32-1024 [10], 10^2-10^3 [13], 10^4 [29]
Catalog Size	800 [70], 10^3 [11], 10^4 [10, 14], 2×10^4 [86], 4.5×10^4 [13], 10^6 [29], $10^8(1PB)$ [15, 16]
Cache Catalog Ratio	10^{-5} [15, 16], 1% [29], 5% [14], 10% [11], 0.22%-2.22% [13], 0.32%-10.24% [10], 0.74%-11.9% [86]
Topology	Cascade Line [17,24,25,29,85,86], Tree [10–12,15,22,70,85–87], Mesh [21, 29], ISP [14–16], Ebone [13], GT-ITM [53], CERNET2 [75], Two-tier [9]

3.3.1 Request Traffic: On-demand vs. Real-time

CCN is designed to better handle and satisfy the Internet traffic generated by end users through the in-network caching property. Therefore, the request traffic is extremely critical in determining the caching performance. Works [54, 55] have confirmed that the Internet web content requests follow Zipf-like distribution, which distribution asserts that the probability of requesting the i^{th} most popular content is inversely proportional to its popularity ranking which is defined as

$$Pr\{C_i\} \propto i^{-\alpha} \tag{3.1}$$

The parameter α is the slope of the log / log representation of the number of requests to the contents as a function of its popularity rank i. Zipf-like distribution describes the web

requests are significantly affected by the content popularity. It indicates the phenomenon that the majority of requests in the network are sent for most popular contents whose amount is very small compared with the total contents in the network.

In our research, we divide the request traffic into two types: on-demand and real-time. On-demand requests are sent for on-demand contents (e.g., email, web page, documents and installation files) that are accessed frequently by large amount of users. On the contrary, real-time requests merely used for requesting time sensitive contents (e.g., online gaming, multimedia streaming and stock trading) that are accessed individually by each user. During the handover process, real-time contents are more vulnerable to QoS degradation and therefore need to be carefully considered. According to the fact the Internet traffic following Zipf's like distribution, we assume the on-demand requests follow Zipf's like distribution while real-time requests are non-duplicated continuous sequence.

Many CCN caching studies [11–16, 29] consider the Zipf-like request traffic with different α range from 0.5 [14] to 2.5 [16], as shown in Table 3.4. Also, [14, 15, 53] use Mandelbrot-Zipf distribution in which the plateau parameter q is used. However, no consensus has been reached on exact traffic model yet. In our work, we use Zipf-like distribution as our traffic request model to match most of the studies. While there is no conclusion on the actual α setting in Zipf-like distribution either. [55] investigates several request traces and shows that these traces follow Zipf-like distribution with different α values range from 0.64 to 0.83 as shown in Table 3.4. But the α value can beyond this range ($\alpha > 2$)for those extremely popular contents such as Youtube videos [88].

Clearly, the range of α parameter is an important factor we need to carefully consider as it will shape the request traffic pattern. Hence, we evaluate the requests probability (from 60% to 99%) for popular contents with α value $\in [0.5, 2.5]$ and catalog size = 10000 in Fig. 3.4. The rank of contents represents the number of objects that satisfy the 60% to 99% user requests. We can see that the rank of contents decreases significantly as α value increases. For instance, when $\alpha = 0.5$, it requires more than 8000 contents to be cached to satisfy 90% requests, while only 3 contents are enough to satisfy the same amount of requests when $\alpha = 2.5$. For this figure, we can expect that the caching performance of CCN is dramatically determined by the popularity parameter α. In our simulation, we first explore a wide range of $\alpha \in [0.6, 1.4]$ and evaluate the effects of α setting on overall CCN performance. After that, we use $\alpha = 0.8$ in our later simulation since it matches the α setting in most of studies and it also corresponds to the phase transition shown in Fig. 3.4.

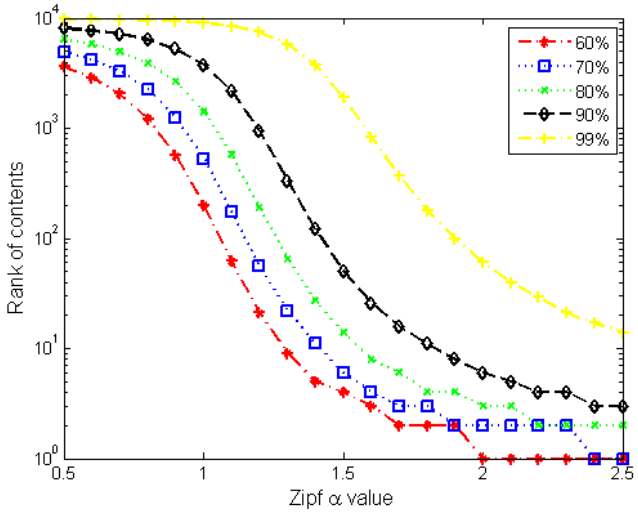


Figure 3.4: Effect of α value on content ranking.

3.3.2 Cache and Catalog Size

From Sec. 3.3.1 we know that α value in Zipf-like distribution drastically control the request pattern. And Fig. 3.4 shows that when $\alpha < 1$, the number of contents satisfying 60% request (or more) is still very large by comparing with the total number of contents (catalog size). This result implies that cache size and catalog size are also critical factors that determine the caching performance in CCN. On one hand, small cache size may limit the CCN content dissemination performance because only a few requested can be satisfied within the network. On the other hand, large cache size will increase the complexity between cache and outgoing interfaces, which is not acceptable under the line speed caching requirement of CCN. In addition, the catalog size should be much larger comparing with cache size as it is the case in current Internet traffic. Hence, reasonable parameter setting about cache and catalog sizes should be seriously considered. Many related work have discussed this issue and used particular cache and catalog size in their evaluation process. We summarize their system parameters in Table 3.4 with Chunk Size, Content Size, Cache Size and Catalog Size respectively (unless otherwise specified, the unit is content).

From the table we can see that most of studies do not consider the packets in chunk level. They argue that the performance of cache policies at content level can directly reflect chunk level case since the goal of cache policies is to improve the hit ratio at CCN nodes. Therefore, it is straightforward that content-level cache policies can be extended to handle the chunk level case, as long as efficient routing protocols and assembling process are used. As most of the studies focus on content level performance, the size of contents varies from 1KB [87] to 10MB [10,15,16]

which is used for YouTube video contents. Some other work simply use content as unit instead of physical storage capacity by assuming the processing delay for content caching and eviction is negligible. This assumption holds as (i)the line speed requirement and (ii) the relatively small packet size in CCN . In our work, we use packet at content level and select the content size to be 1 as most of the studies [11, 13, 14, 29].

As mentioned above, cache size at CCN nodes and catalog size are two important factors that can significantly affect the performance of cache policies. The size of cache in related work starts from 100KB [87] to 16GB [86] or 100 [11] to 10^4 [29] at content unit. On the other hand, the catalog size is set at the range from 800 [70] in a small scale simulation to 10^8 [15,16] for the largest YouTube video diffusion system. Although we know that, comparing with catalog size, the cache size should be much smaller, there is no consensus on the exact value of cache size and catalog size so far. To this end, we use Cache-to-Catalog Ratio (CCR)as a new parameter to explore and investigate the system setting. We expect that this CCR is more intuitive to determine the overall caching performance by matching it to the α effect in Fig. 3.4. For example, at the case that $\alpha = 1$ and perfect popularity-oriented cache policy is used, the CCR equals to 2% can satisfy around 60% requests while the ratio has to increase to 6% to satisfy 70% requests. We also summarize the CCR of related work as shown in Table 3.4. The ratio used in these studies varies between 0.001% [15, 16] to 11.9% [86]. In our work, we will evaluate the effect of cache size to the cache policies and therefore we use CCR at the range between 0.5% to 8%. As far as the catalog size is concerned, we select the size to be 10^4 which is the same magnitude used in most studies [10, 13, 14, 86].

3.3.3 Network Topology

Although many CCN studies have conducted simulations to analyze the CCN performance or evaluate their designs, most of them are limited to either cascade line or binary trees network topologies. We summarize the network topologies used in these works in Table 3.4. Simple network topologies used in simulations can minimize the uncertainty of CCN caching process. But it can be a problem as some designs will take the advantage of pre-defined simple topology. For instance, [12] proposed its path capacity based caching scheme under the assumption of simple tree topology and proved that their scheme performs well to eliminate redundant traffic and improves caching performance in CCN. However, we cannot tell whether this scheme can bring the same benefits in mesh network topology which is not distributed in hierarchical manner. Studies such as [24, 25] proposed mobility support schemes based on the cascade line topologies in which pre-requested contents can be cached in the same path. But this assumption cannot hold in tree topology if user moves from one leaf node to another and there is no common path between these two. Meanwhile, complicated topology such as different ISP scenarios [14–

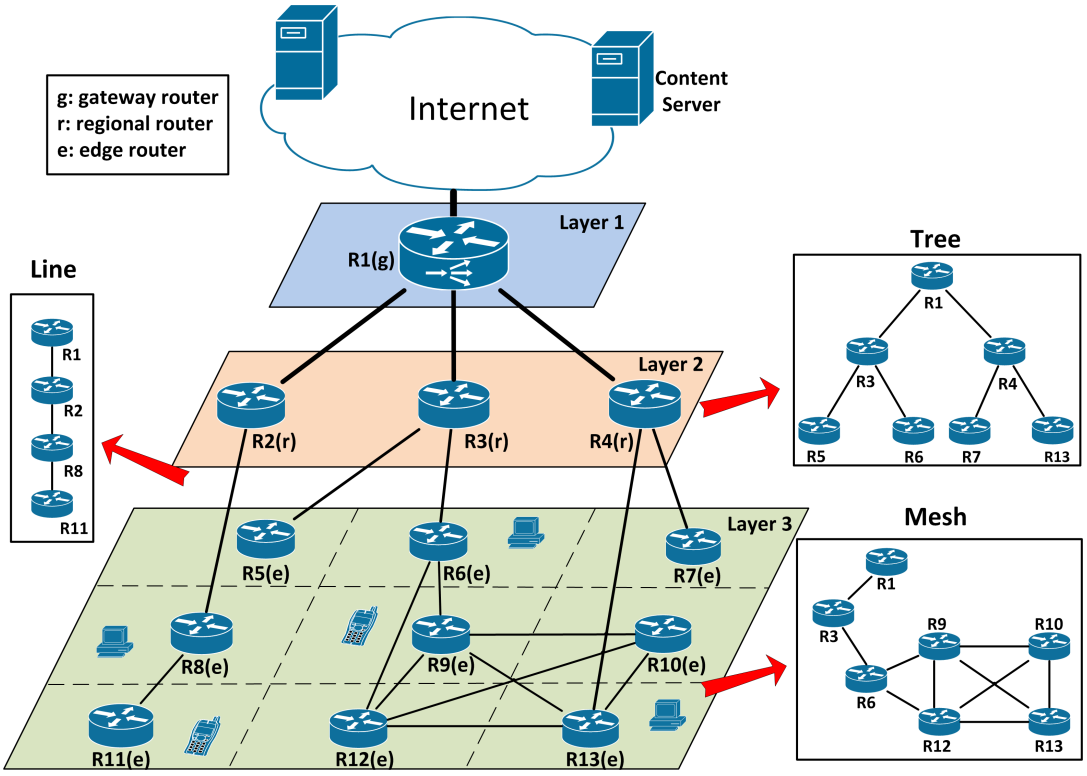


Figure 3.5: Three layers hybrid network topology.

16] might blur the effectiveness of cache policies since they are evaluated in average level. In order to fully utilize the in-network caching property, it is important to target different topologies with specific caching schemes.

In our work, we design a three layers hybrid network topology (shown in Fig.3.5) combining cascade line, mesh and tree topologies. At the top layer, one gateway router is deployed to connect the Internet (content servers). If content cannot be found within this network, the *Interest* packets will be forwarded to this gateway router. At the second layer, three regional routers are placed to connect the top level and bottom level. And at bottom layer, nine edge routers are uniformly distributed in $3km \times 3km$ area and they are directly connected with aggregated end users. We also assume that these edge routers support wireless connectivity so that this topology can be used in mobile scenario as well. All types of routers have in-network caching capability with the same cache size in default. By using this hybrid network topology, we are able to evaluate the performance of cache policies under four scenarios (line, tree, mesh and hybrid) and we believe the results came from this network topology are more comprehensive.

Table 3.5: CDP and CRP used in simulation

Cache Decision Policy	***CacheAll***: cache every arrived content
	Rdm(P): cache content randomly with the fixed probability $P = 0.5, 0.75$
	Betw: cache content at nodes that have higher betweenness centrality
	ProbCache: caching decision based on the path
Cache Replacement Policy	***LRU***: least recently used content will be replaced

3.3.4 Cache Decision and Replacement Policy

As discussed in Sec. 2.1, caching decision policy (CDP) in CCN is an area that has not been fully studied yet. Current CCN simply uses universal caching ($CacheAll$) policy via which every arrived content needs to be stored in the cache. However, some researchers argue that such indiscriminate cache policy is inadequate to handle current Internet traffic which mainly follow Zipf-like distribution. Hence, [11,12] proposed more intelligent, adaptive caching decision policies for CCN. In our evaluation process, we use the performance of default caching decision policy ($CacheAll$) as the baseline. Random cache ($Rdm(P)$) is used in our simulation as it shows good performance in [11, 15]. We also implement $Betw$ [11] and $ProbCache$ [12] for the caching performance comparison. Through the simulation in our hybrid network topology, we can get a better understanding of how topology-assistant caching decision policies work. As far as the cache replacement policy, we only use LRU in our simulation because (i) it is the default caching replacement policies in CCN prototype [4], (ii) it is widely used in related works for CCN caching performance evaluation and (iiI) it is simply but effective for handling the Zipf-like request traffic. We list all CDP and CRP used in our simulation in Table 3.5.

3.3.5 System Parameters

Table 3.6 lists all the system parameter notation and values used in our simulation. In the simulation, we assign one aggregated user for each router. During the simulation process, aggregated user generates request traffic following R_d and the total request for each aggregated user is R_t. R_r denotes the real-time requests over total requests ratio. When $R_r = 0$, request traffic is pure on-demand traffic.

Table 3.6: System parameters notation.

Parameter	Meaning	Values
R_d	Request traffic distribution	Zipf-like distribution
α	Zipf exponent	[0.6, 1.4]
T	Network topology	line, tree, mesh, hybrid
R_t	Request from aggregated users	10^5
R_r	Request rate	25Hz
P	Packet size	1
c	Cache size	[50, 800]
C	Catalog size	10^4
$\frac{c}{C}$	Cache-to-Catalog ratio	[0.5%, 8%]
CDP	Cache decision policy	$CacheAll$, $Rdm(P)$ $Betw$, $ProbCache$
CRP	Cache replacement policy	LRU

3.4 Simulation Results and Discussion

3.4.1 Performance Metrics

Caching performance are mostly evaluated by Cache Hit Ratio, which demonstrates the probability of one request can be satisfied at the cache node. However, Cache Hit Ratio at individual node only shows the caching performance locally. It cannot directly provide a higher view of overall cache performance in CCN network. CCN implements in-network caching in network level for improving the content delivery performance. Therefore, additional performance metrics are needed to interpret the overall cache performance at network-wide perspective. More importantly, at the users perspective, the latency of contents retrieval is one of the most concerning factors. With the assumption that perfect link condition, we measure the hop count that packet travels to demonstrate content delivery latency.

In our simulation, we provide the usual Cache Hit Ratio as the common metric for single cache performance evaluation. Besides, we define new metric: Cache Diversity and Hop Distance Ratio for node level and network level metrics respectively.

I. Cache Hit Ratio

Cache Hit Ratio is the most commonly used metric in caching performance evaluation as it directly reflects the percentage of requests that can be satisfied at the cache. The general Cache Hit Ratio is obtained from Eq.(3.2), which can vary in the interval [0,1], with the lower bound achieved in the worst case that no request is satisfied at the cache. N_h represents the number of cache hit and N represents the total number of requests.

$$Cache\ Hit\ Ratio = \frac{N_h}{N} \tag{3.2}$$

II. Content Diversity

The Content Diversity, defined in Eq. (3.3), investigates the content popularity distribution within the cache. We classify the top popularity range $\in [1, C_{size}]$, the second popularity range $\in [C_{size}+1, 2C_{size}]$, and so on ($C_{size}$ denotes the cache size). For example, a content with popularity rank at $No.55$ belongs to first range if $C_{size} = 100$ or second range if $C_{size} = 50$. Only first three popularity ranges are monitored in our simulation. $|C_i|$ in Eq. (3.3) denotes the number of contents fall in i_{th} popularity range. Content Diversity results are collected in every 1000 arrival requests during the simulation.

$$Cache\ Diversity(i) = \frac{|C_i|}{C_{size}} \quad i = 1, 2, 3. \tag{3.3}$$

III. Hop Distance Ratio

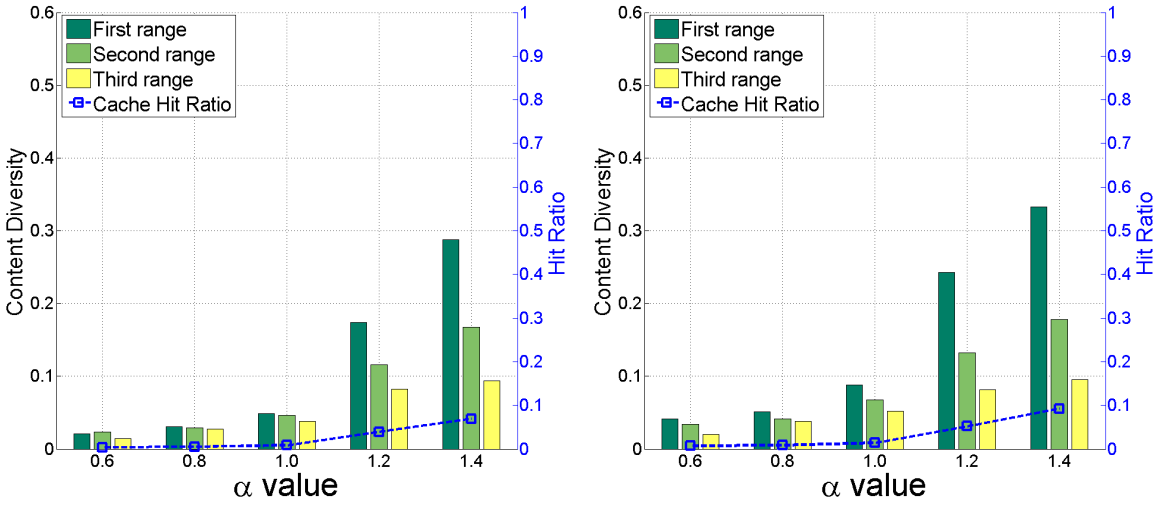
In user's perspective, the latency of content retrieval is the most important factor that impact the QoS experience. In order to lower the content delivery latency, contents should be cached close to the end users so that they can be fetched faster than from the content server. We define user-centric metric Hop Distance Ratio in Eq.(3.4) which is the average ratio of actual hop count h_i that packet travels over the total hop count H_i from user to content server.

$$Hop\ Distance\ Ratio = \frac{1}{N}(\sum_{1}^{N} \frac{h_i}{H_i}) \tag{3.4}$$

where h_i is the hop count of *Interest* for i^{th} content from user to the first router that content hit occurs. H_i is the hop count from the user to the original content server for i^{th} content. N is the total number of requests. Notice the Hop Distance Ratio is normalized metric that varies from 0 to 1, with the lower bound as the best case that user can reach the content at direct connected router. In addition, lower Hop Distance Ratio value means more bandwidth saving. In non-caching network, as all the requests has to be forwarded to the content server, the Hop Distance Ratio = 1.

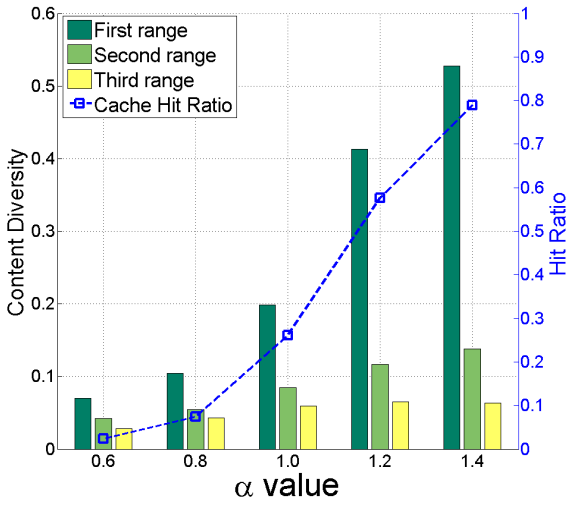
3.4.2 Impact of α Value in Request Traffic

We first evaluate the impact of α value in request traffic with *CacheAll* decision policy and *LRU* replacement policy in hybrid topology scenario. We vary the α value from 0.6 to 1.4 with Cache-to-Catalog Ratio (CCR)=1%. Fig.3.6 plots the Cache Hit Ratio and Content Diversity



(a) Gateway router

(b) Regional router



(c) Edge router

Figure 3.6: Hit ratio and content diversity results with varying α (CCR= 1%).

of routers distributed in three layers. Cache Hit Ratio values are the average values of router(s) in each layer.

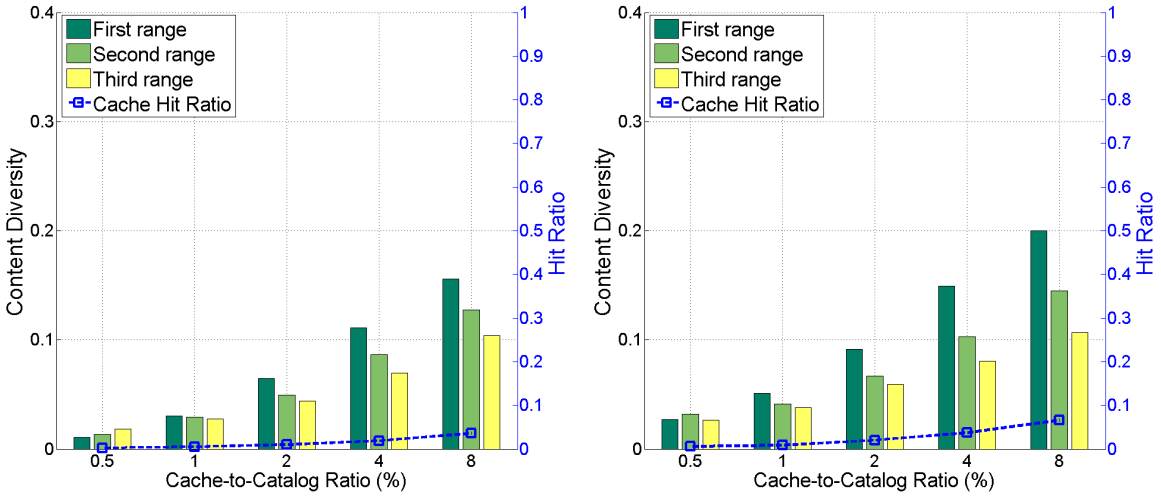
As expected, hit ratio performance is improved with increasing α value. However, the rising rates among three layers are very different. Hit ratio at edge routers which are directly connected with end users rises dramatically from 0.02 to 0.8 as α increases, especially after $\alpha > 0.8$ (as shown in Fig. 3.6c). On the contrary, in both regional and gateway routers, the hit ratio performance increase steadily from 0.01 to 0.1 as α varies from 0.6 to 1.4. Due to this significant

hit ratio performance difference within three layers, the overall hit ratio for entire network is neutralized.

To further explore how α value affects the category of contents stored in the caches, we demonstrate the average Content Diversity in all three types of routers as shown in Fig. 3.6. We observe that, with increasing α, the percentage of cached contents in top three popularity ranges increases accordingly, leading to higher hit ratio results. For instance, over 50% of cached contents in edge router belong to top popularity range in $\alpha = 1.4$ scenario, with hit ratio result up to 0.8. However, regional and gateway routers are not the same case. Although 33% cached contents belong to top popularity ranges in regional router at α =1.4 scenario, the hit ratio barely reaches 0.1. This is because duplicated contents are cached along the routing path due to *CacheAll* policy. By adopting *CacheAll* policy, every content routed back from upper level (e.g., content server) is stored repeatedly in all caches along the delivery path. These duplicated contents, to some extend, are cache pollution in upper layers since they occupy the cache space but cannot satisfy most of the forwarded requests came from lower layers, leading to poor caching performance. Ideally, in hybrid topology, regional and gateway routers should cache contents belong to second (or third) popularity range so that to satisfy most of the arrival requests forwarded from lower layer routers.

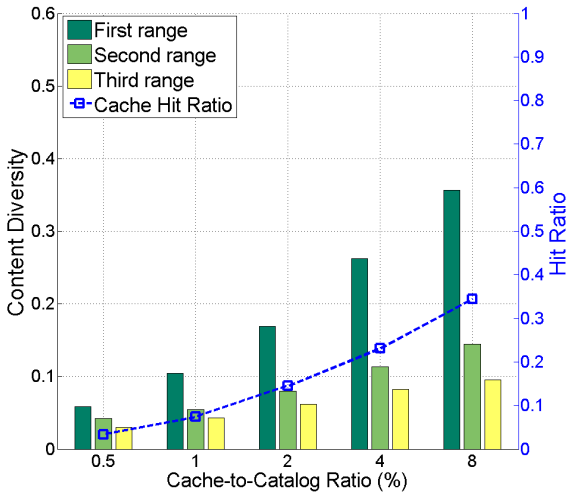
From the results demonstrated in Fig. 3.6, we find that the α value has significant impact on the in-network caching performance in CCN. The α value in request traffic determines the request frequency among contents and popular contents are accessed more frequently as α value becomes larger. This implies that in a large α value scenario, caching one popular content can be much more efficient than caching hundreds of unpopular contents in both caching performance and cost efficiency perspectives. However, we also observe that caching the most popular contents is not always the best strategy for every routers. Simulation results show that the gateway and regional routers provides very limited contribution on content retrieve though they have the same cache size as edge routers, leading to low caching performance in network level. This cache pollution introduced by *CacheAll* should be avoided in routers whose request traffic is modified by downstream routers. In another word, content popularity at each router is evaluated by relative content popularity based on the arrival request traffic instead of absolute content popularity for the entire catalog. We summarize our findings in this simulation as follows:

Remark 1 *Content popularity (α value of request traffic) has direct effect on in-network caching efficiency in CCN. Keeping popular contents in the cache improves the caching performance in CCN. However, caching the most popular contents is not always the best strategy. Instead, caching decision making should depend on arrival requests.*



(a) Gateway router

(b) Regional router



(c) Edge router

Figure 3.7: Hit ratio and content diversity results with varying CCR ($\alpha = 0.8$).

3.4.3 Impact of Cache Size

Besides α value, CCR is another important factor that affects the in-network caching efficiency in CCN. With the same environment settings as previous simulation, We assess the Cache Hit Ratio and content diversity respectively for varying settings of CCR from 0.5% to 8% with fixed $\alpha = 0.8$ as it is close to the average α value obtained from real Internet traffic trace [55]. Notice that cache size under CCR=8% is extremely large considering the huge amount of Internet

traffic.

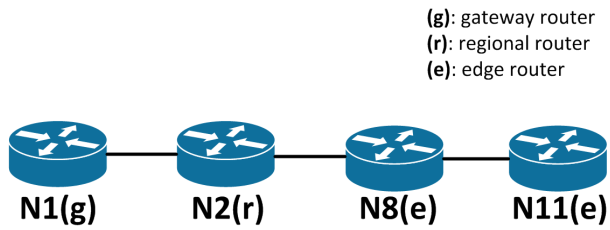
The simulation result is demonstrated in Fig. 3.7. Clearly, we can observe that larger CCR introduces higher hit ratio for router(s) in all three layers. Again we find that the increasing rates among three layers are different. To our surprise, the effect of cache size expansion on caching efficiency improvement is not as considerable as what we expect. And by comparing with Fig. 3.6, we can easily tell that the effect of CCR on caching efficiency is not as significant as α value. For instance, for routers in layer three, even though we doubly expand the cache size from CCR=0.5% to 8%, the hit ratio results increase slowly from 0.02 to 0.35. The overall hit ratio performance at CCR=8% is around 0.15.

Also in Fig. 3.7, we can inspect the content popularity distribution in different layers. The same as previous α value simulation, the compositions in each layer are similar: the unpopular contents are the majority in the cache followed by first range contents, leading to low caching efficiency, especially for routers in layer one and layer two. This result again confirms that *CacheAll* policy is not effective to target and cache the *correct* contents for improving the in-network caching efficiency in CCN. For example, in layer three, most of the caches are used for caching unpopular contents which barely contribute to hit ratio performance. From the results, we realize that simply increasing the cache size is not efficient for improving caching efficiency in CCN. In addition, large cache size results in higher expense, processing delay and computing complexity which are not feasible in CCN. To avoid such problems, routers should be able to intelligently cache the most valuable contents under limited cache size. We summarize our findings in this simulation as follows:

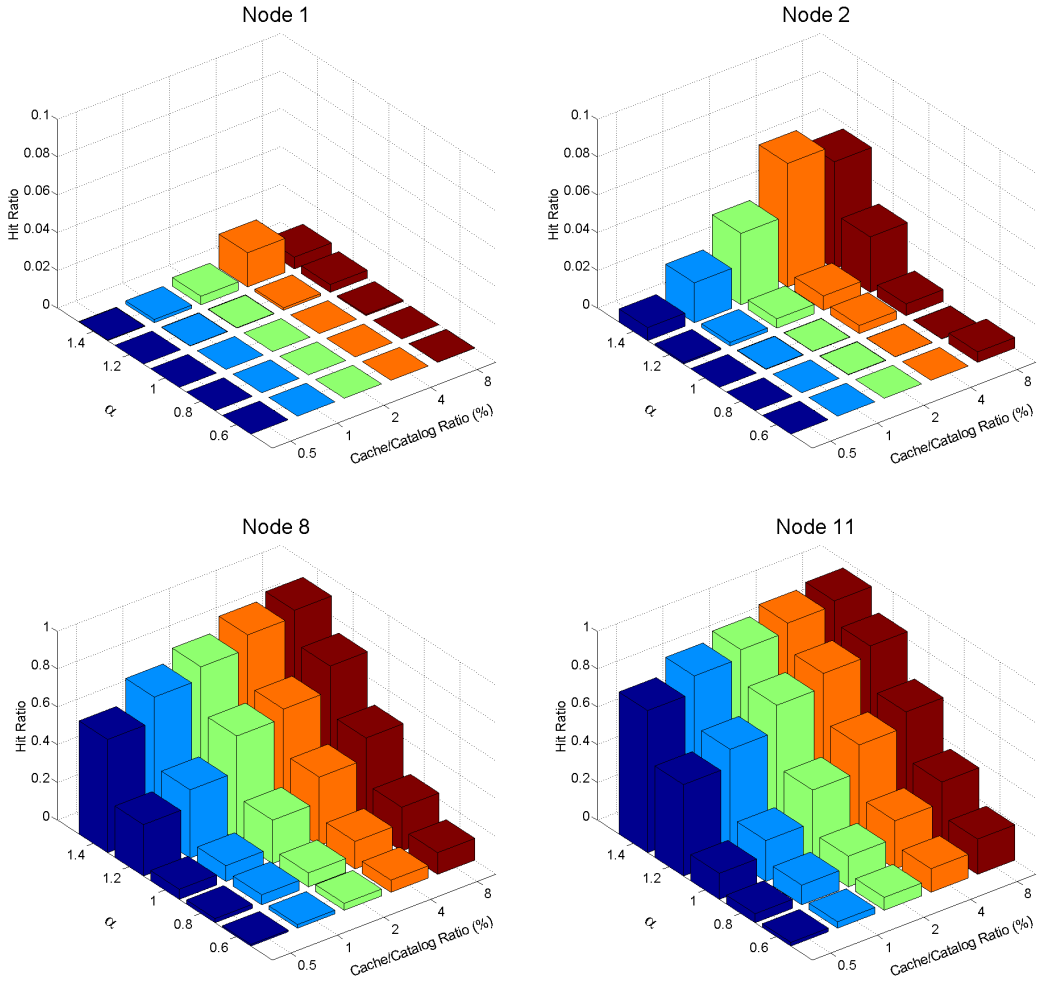
Remark 2 *The impact of cache size is limited. The results imply that simply expanding the cache size is not cost efficient to enhance the caching efficiency in CCN. Moveover, considering the rapidly increased information volume in the Internet, expanding the cache size in router is not practical.*

3.4.4 Impact of Network Topology

Previous α and CCR simulations are conducted in hybrid topology scenario, in which we realize that location and connection among routers are also critical factors that affect the caching performance in CCN. To understand the effect that router location and organization bring about, we conduct another group of simulations with four topologies and they are (a) line topology (b) tree topology (c) mesh topology and (d) hybrid topology. In each topology, there are one gateway router, one or more regional router(s) and multiple edge routers. Only the edge routers directly receive request traffic from aggregated end users. And all the unsatisfied requests will be forwarded to gateway router which connects to the content servers out of the subnetwork. In this group of simulations, we evaluate the hit ratio at edge routers with full



(a) Line topology



(b) Hit ratio results of nodes

Figure 3.8: Hit ratio results in line topology with full range of α and CCR value.

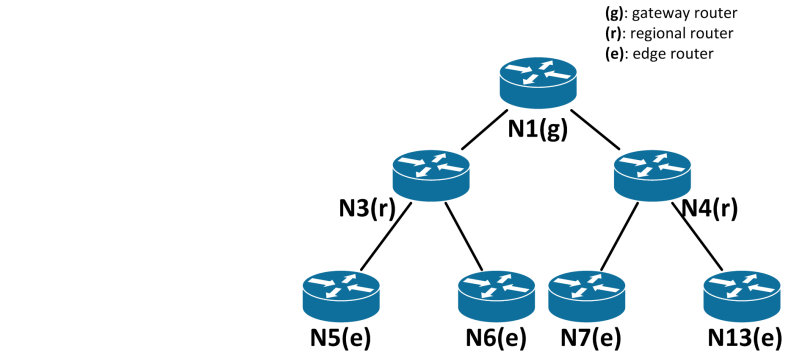
range of α value from 0.6 to 1.4 and CCR from 0.5% to 8%. We believe that such wide range settings can provide comprehensive results for us to analyze the impact of topologies on caching efficiency. The same as previous simulation, we consider *CacheAll* as cache decision policy and

LRU as cache replacement policy and shortest path routing.

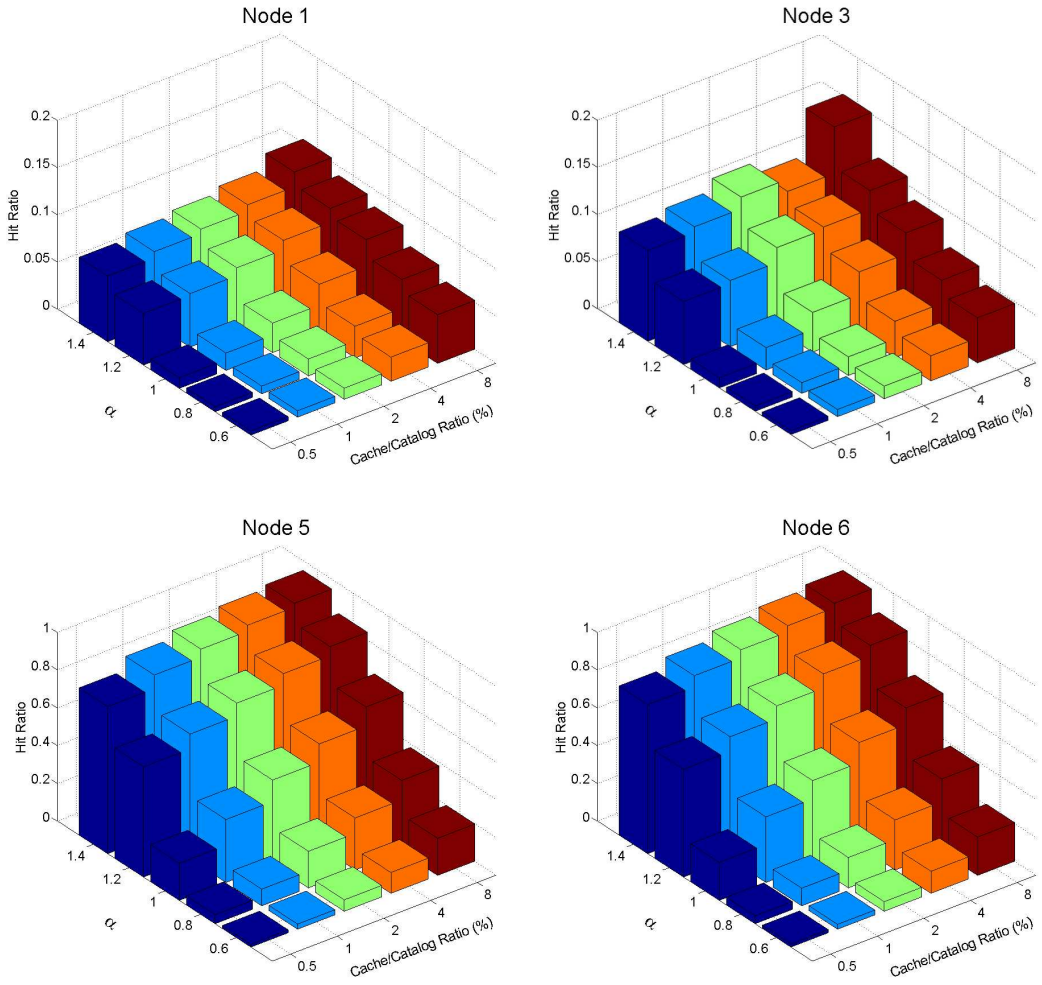
I. Line Topology Scenario

The first topology is the cascade line topology in which routers are connected one by one as shown in Fig. 3.8a. Node 1 is the gateway router and node 2 is the regional router. Both node 8 and node 11 are edge routers with end users connected directly. As the results demonstrated in Fig. 3.8b, we can observe that node 1 and node 2 provides very poor hit ratio performance. Even in the case that α is 1.4 and CCR is 8%, the hit ratios of these two routers remain below 0.06. On the contrary, the hit ratio in node 8 and 11 are much higher compared with node 1 and node 2 that a few hops away. As the same result shown in previous α and CCR evaluation, increasing either α value or CCR can improve the caching performance. We can also confirm the findings that caching performance is more sensitive to α value change. Although node 2 and node 8 are one hop away from each other, the caching performance are dramatically different. The reason of this phenomenon is that , as a edge router, node 8 receives request directly from end users while node 2, as a regional router, only handles the forwarded requests that cannot be satisfied at edge routers. The edge router acts like a filter that blocks (satisfies) most of the user requests for popular contents that are stored in its cache. For those unsatisfied requests, edge router passes (forwards) them to non-edge router which is located at the higher layers of the network and can not receive user requests directly. With limited cache size, non-edge routers are less likely to store the content to satisfy most of the forwarded requests after the filter since the majority of this requests are looking for less popular contents, leading to low caching performance in node 1 and node 2.

In fact, *CacheAll* policy in single line topology aggravates the performance on node 1 and node 2. The algorithm of *CacheAll* policy adopted in routers along the routing path cache every arrival content indiscriminately. And single line topology means that only one routing path existed in the network. Therefore, contents forwarded by upper level nodes (e.g., content server) are cached duplicately in the intermediate routers along the path. Duplicated contents cached in non-edge routers have less chance to be accessed as the requests for these contents are satisfied by edge routers. More importantly, most of the contents that travel through non-edge routers belong to the tail of the request, which means that they have lower probability to be accessed again. Thus, contents cached in non-edge router (node 1 and node 2) cannot improve the caching performance in this line topology case. In addition, by carefully inspecting the results in Fig.3.8b, we can find that hit ratio performances decreases along the routing path from edge router endpoint (node 11) to the gateway router endpoint (node 1). Although both node 8 and node 11 are edge routers, the hit ratio in node 11 outperforms that in node 8. The reason of such difference is that node 8 caches contents not only requested by its end users but



(a) Binary tree topology



(b) Hit ratio results of nodes

Figure 3.9: Hit ratio results in tree topology with full range of α and CCR value.

also contents requested by lower level router (node 11). These passing contents, to some extend, pollute the cache in node 8 since they are less popular ones, resulting to the degradation of caching performance in node 8.

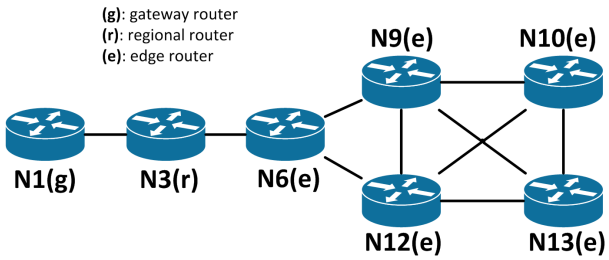
II. Tree Topology Scenario

The second topology we evaluate is binary tree topology. The router organizations in tree topology is shown in Fig. 3.9a. Node 1 is the gateway routers and both node 3 and 4 are its leaves which act as regional routers. Node 5, 6, 7 and 13 are leaves of node 3 and 4 respectively. All of them are edge routers that directly connect with end users. Since tree topology is symmetric and for simplicity purpose, we only demonstrate the results of left branch (Node 1,3,5 and 6). Hit ratio result of these four routers are plotted in Fig.3.9b. From this figure, we can see that , as all the edge routers are located in the endpoint of the routing path, their overall performances under different α and CCR settings are optimized because they don't suffer the cache pollution problem that we mentioned before. The most significant differences of the results between line and tree topology are the caching performance improvement in node 1 and node 3. By comparing with the routers at the same location in line topology (node 1 and node 2), we can see a significant caching performance boost in node 1 and node 3 under full range of α and CCR values. In tree topology. both node 1 and node 3 have more than one connection at their downstream path. Such connections imply that contents cached at these two routers have multiple end destinations, making less duplicated contents are cached along the path. For example, as node 3 connects with both node 5 and node 6, the cached contents that requested by node 5 can be accessed by node 6 later. This also explains the performance improvement in node 1 which located on four different routing pathes.

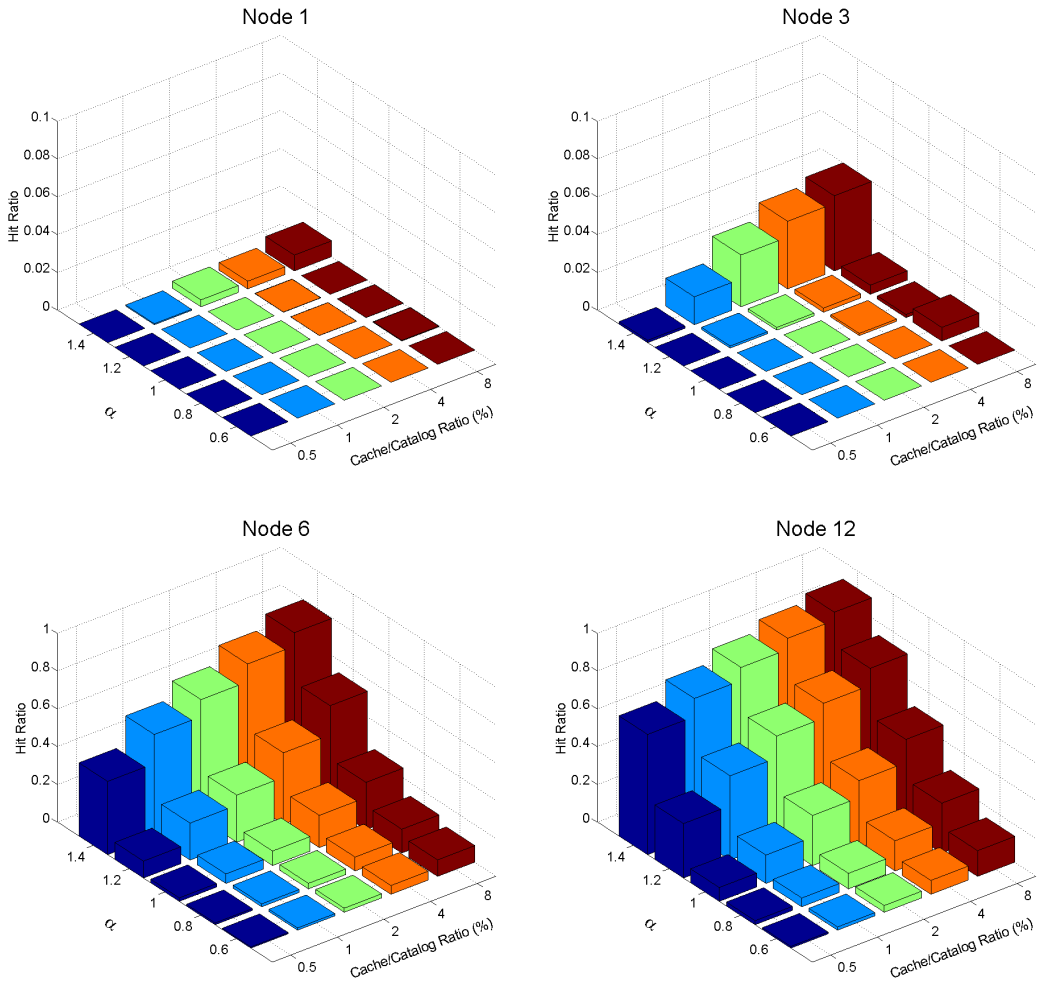
However, the caching performance at node 1 and node 3 are still relative low comparing with node 5 and 6 that located at the endpoint of the routing path. As the intermediate routers in tree topology, the caches in node 1 and node 3 are polluted by their leaves. In summary, the caching performances of non-edge routers in tree topology scenario are improved, implying that nodes laid on multiple pathes can avoid duplicated contents to be cached along each path. The same as line topology, caching pollution problem still remains unsolved as *CacheAll* policy is used, leading to serious caching efficiency problem.

III. Mesh Topology Scenario

Mesh topology is the third topology we used in our simulation. As shown in Fig. 3.10a, node 1 and node 3 are gateway router and regional router respectively. The rest routers are all edge routers, among which node 9, 10, 12 and 13 are connected with each other and router 9 and 12 receive forwarded requests from node 10 and 13 with the same probability. Different with tree



(a) Mesh topology



(b) Hit ratio results of nodes

Figure 3.10: Hit ratio results in mesh topology with full range of α and CCR value.

topology scenario, in which all edge routers are located at the endpoint of the routing path, node 6, 9 and 12 are intermediate edge routers at two layers in mesh topology scenario. Except

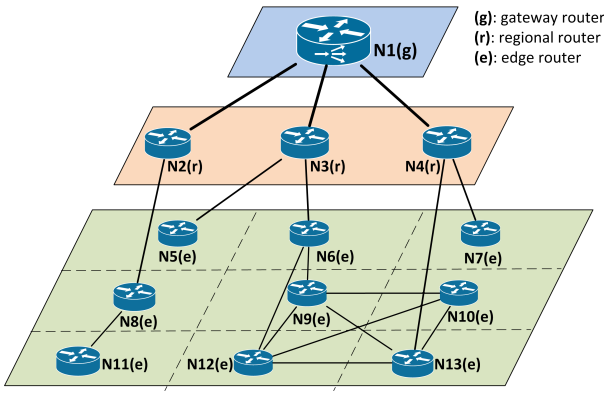
for the topology change, all other settings remain the same as previous simulations. Due to the symmetry of mesh topology, node 9 and 10 perform the same as node 12 and 13. Meanwhile, edge routers node 10 and 13 only receive request from their own end users which are exactly the same as previous simulations. Hence, we only plot the hit ratio results of four routers as shown in Fig. 3.10b. For performance comparison between endpoint edge routers and other routers, please use the result of node 5 in Fig. 3.9b or node 11 in Fig. 3.8b.

First of all, we can see the hit ratio results of different routers are similar with the performance in line topology case. The overall hit ratio decreases from edge routers end of the routing path to the gateway router. Again, we observe poor performance in node 1 and 3 as they form the single line topology with node 6. Because of duplicated content stored in node 1 and 3, less forwarded request from node 6 and its leaves can be satisfied. This result implies that router organization at the lower end of the routing path doesn't affect the caching performance of routers located in the middle of line section of the path. Surprisingly, we see that node 6 has serious degradation in hit ratio though it is edge router with multiple leaves connected in its downstream. This is because the negative effect of caching pollution outweigh the positive that multi-sources connections bring about. As all the requests that cannot be satisfied at lower level routers are forwarded to node 6, the cache of node 6 is updated faster. *CacheAll* policy adopted in node 6 replaces popular contents with less popular contents that are requested by lower level routers. Finally, by inspecting the hit ratio at node 12, we can also see the performance degradation comparing with edge routers which are not located at the middle of the routing path. From this simulation, we can conclude that reducing the connections between edge routers can minimized the cache pollution problem when *CacheAll* policy is used.

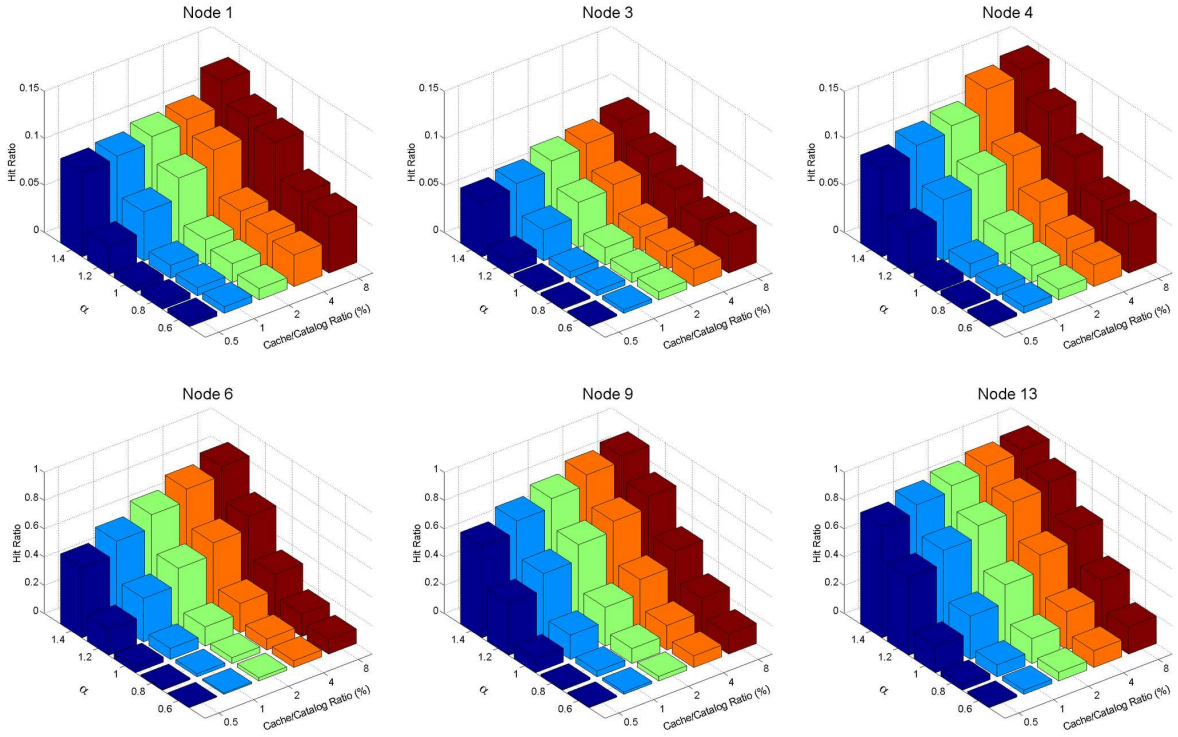
IV. Hybrid Topology Scenario

After evaluating line, tree and mesh topology individually, we combine these three topologies together and form the hybrid topology as shown in Fig. 3.11a. In hybrid topology, we place one gateway router, three regional routers and nine edge routers. The purpose of hybrid topology simulation is to examine whether our findings in previous individually simulations still hold. In hybrid topology, node 5, 7, 10, 11, 13 are all endpoint edge routers and thus we only plot the result of node 13. We don't show the results of node 2 and 8 since they are similar with their performance in line topology simulation conducted before. Node 12 acts the same as node 9 since both of them are intermediate routers receiving requests from node 10. And the hit ratios of the rest routers are demonstrated in Fig. 3.11b.

By inspecting the hit ratio, we can see that the caching performance of node 1 in hybrid topology is better than the result in line and mesh topology scenarios while close to the performance in tree topology. This is because node 1 has three downstream connections and



(a) Hybrid topology



(b) Hit ratio results of nodes

Figure 3.11: Hit ratio results in hybrid topology with full range of α and CCR value.

therefore less duplicated contents on the single path. When a request forwarded by regional routers arrives, it has higher probability to find the contents that are requested by other regional routers before. The same behavior can be found at node 3 which connects one more edge router comparing with the mesh topology scenario.

As expected, by comparing hit ratios between node 3 and 4, we can see that node 3 with

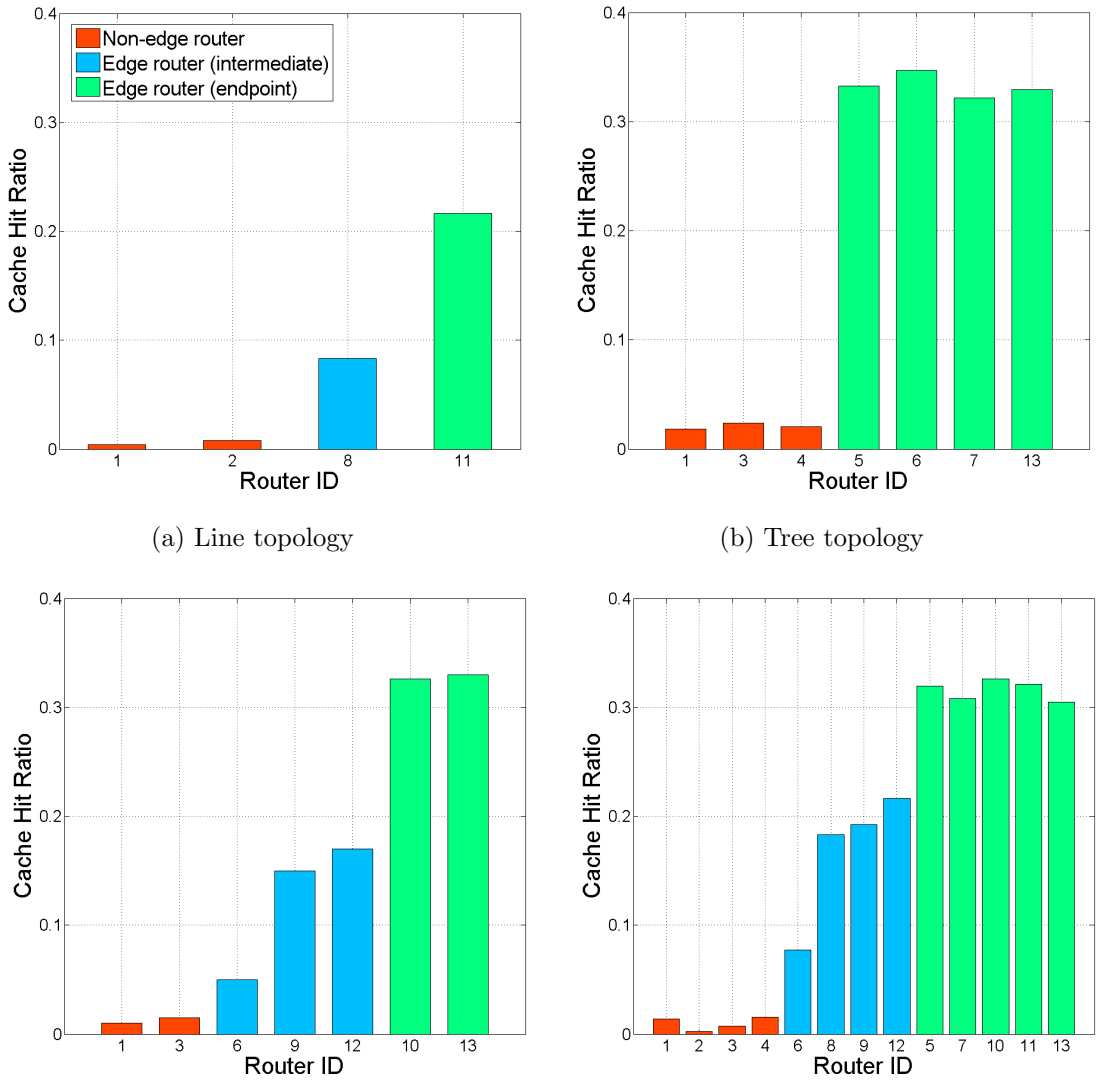
more downstream connections has lower hit ratio. The reason of this result is the same in mesh topology as more unpopular contents are cached due to the *CacheAll* policy algorithm. At this situation, *LRU* replacement policy cannot work properly to keep the right contents in the cache. And the performance is even worse at small α and CCR case, with which contents in the cache are flushed more frequently. As for node 6, 9 and 13, we can still observe the same behavior that they suffer less cache pollution as they locate close to the end of the routing path.

V. Discussion

In order to provide a more clear view on the impact of network topology and routers interconnection on CCN, we collect the Cache Hit Ratio results of all routers in four different topologies with $\alpha = 1$ and CCR=1% and present them in Fig. 3.12. First of all, in all topologies, we can observe that non-edge routers (R1 - R4) provides very poor hit ratio performance. On the contrary, the hit ratio in edge routers are much higher especially for endpoint edge routers that are located at the end of the path such as R11 at line, R5 in tree and R13 in mesh topology. This can be explained by analysing the filter effect: edge router acts like a filter that blocks (satisfies) most of the user requests for popular contents that are stored in its cache. For those unsatisfied requests, edge router passes (forwards) them to non-edge router which is located at the higher layers of the network and can not receive user requests directly. Due to limited cache size and the long tail in Zipf-like request pattern, these forwarded requests are less likely to be satisfied in non-edge routers.

In addition, we also observe that edge routers perform differently due to their interconnections. Endpoint routers located at the end of the routing path (e.g., R5, R11 and R13 in hybrid) reach the highest hit ratio, while intermediate routers (e.g., R6 and R8 in hybrid) located on multiple routing paths suffer serious hit ratio degradation. For instance, R6 in hybrid topology provides much lower hit ratio comparing with R5. This is because the arrival request pattern at R6 is changed by combining the direct requests from end users and forwarded request from lower level router (R9, R10 and R12). By adopting *CacheAll* policy, those passing contents, to some extend, pollute the cache in R6 since they are less popular, resulting to the cache performance degradation in R6. This pollution effect is more obvious in line topology scenario as duplicated contents are cached along the single routing path. Based on the results and the analysis above, we summarize our findings as below.

Remark 3 *Edge routers that receive direct user requests are more caching-efficient than non-edge routers in CCN, which again emphasizes the importance of arrival requests. By applying universal caching (CacheAll), routers located on multiple routing path suffer serious cache pollution from downstream. In addition, among four different topologies, line topology performs the worst as it suffers the most serious caching pollution problem.*



(a) Line topology

(b) Tree topology

(c) Mesh topology

(d) Hybrid topology

Figure 3.12: Hit ratio and content diversity results with varying CCR ($\alpha = 0.8$).

3.4.5 Impact of Cache Decision Policy

In previous simulations, we apply *CacheAll* and *LRU* for cache decision and replacement policy in each router respectively . The results from previous simulation show that *CacheAll* policy provides poor performance especially in low α or CCR value scenarios. In addition, the same contents are cached duplicately in the path, leading to low cache utilization efficiency at line topology scenario. In this section, we conduct a set of simulations for evaluating different cache

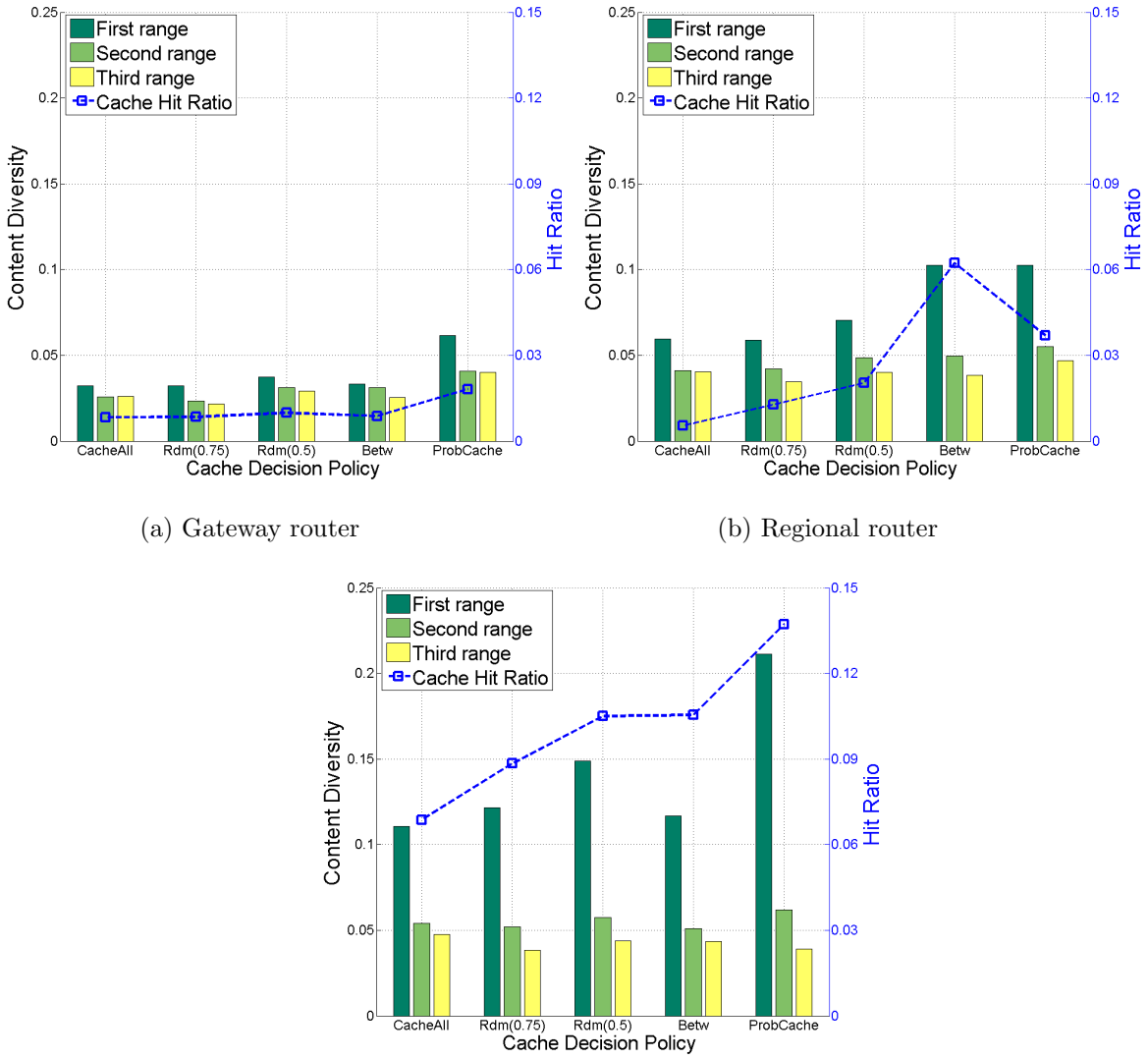
policies as introduced in Sec3.3.4.

Besides *CacheAll*, $Rdm(P)$ is another light-weight decision policy since the caching decision is simply based on a pre-defined probability value. We evaluate the $Rdm(P)$ policy with $P = 0.5$ and $P = 0.75$. Actually, *CacheAll* policy can be considered as a special case of $Rdm(P)$ with $P = 1$. *Betw* is a new policy proposed in [11] for improving the caching performance in CCN. Instead of cache contents at each router, *Betw* utilizes the betweenness centrality of routers located on the routing path and only cache the content at the router which has the highest betweenness centrality value. According to *Betw* algorithm, only node 1, 2, 3, 4, 6, 8 are allowed to cache contents in hybrid topology. We also evaluate another decision policy named *ProbCache* which is introduced in [12]. The idea of *ProbCache* is to fairly allocate cache space for multiplex contents of different flows in caches along a shared path. *ProbCache* makes the caching decision based on caching capability of a path and distance to the destination.

For simulation settings, we utilize hybrid topology as it combines line, tree and mesh topologies. The α value of traffic request is 0.8 and the CCR value is 1%. For results analysis, we apply three performance metrics: Hit Ratio, Content Diversity and Hop Distance Ratio. Hit Ratio and Content Diversity metrics are used to demonstrate the performance at node level while Hop Distance Ratio is used for network level evaluation.

Fig.3.13 shows the result of Hit Ratio and Content Diversity for different cache decision policies among three different router types. First of all, we can see that *CacheAll* policy is the worst decision policy as it provides almost the lowest hit ratio in every router type. By comparing *CacheAll* and $Rdm(p)$, we observe that the Hit Ratio increases as the probability value p decreases (*CacheAll* can be viewed as $Rdm(1)$). This interesting result implies that caching less is a simple way to improve the Hit Ratio at routers. The reason is that by reducing the probability (frequency) of content caching, less popular contents are evicted accidently due to the full cache size. Among all these decision policies, *Betw* is the one that only cache contents at particular routers. The result shows that *Betw* performs the best at regional routers area and the second at edge router. Notice that in *Betw* decision policy, only node 6 and 8 are evaluated in edge router area. Although routers with highest betweenness centrality simply use *CacheAll* policy, *Betw* allow routers to collect more forwarded requests from the other nodes and therefore re-shapes the request traffic distribution, leading to a relative high Hit Ratio performance. When come to *ProbCache* decision policy, we can see that it provides higher Hit Ratio at gateway and edge routers comparing with the others. From user's perspective, higher Hit Ratio at edge router means more contents can be accessed directly or in a shorter distance. To this end, we consider *ProbCache* is slightly better than *Betw* and it is the best decision policies so far.

By inspecting the Content Diversity, we find that at gateway router, the proportions of content in three ranges are similar and total of them occupy about 10-15% of cache space.



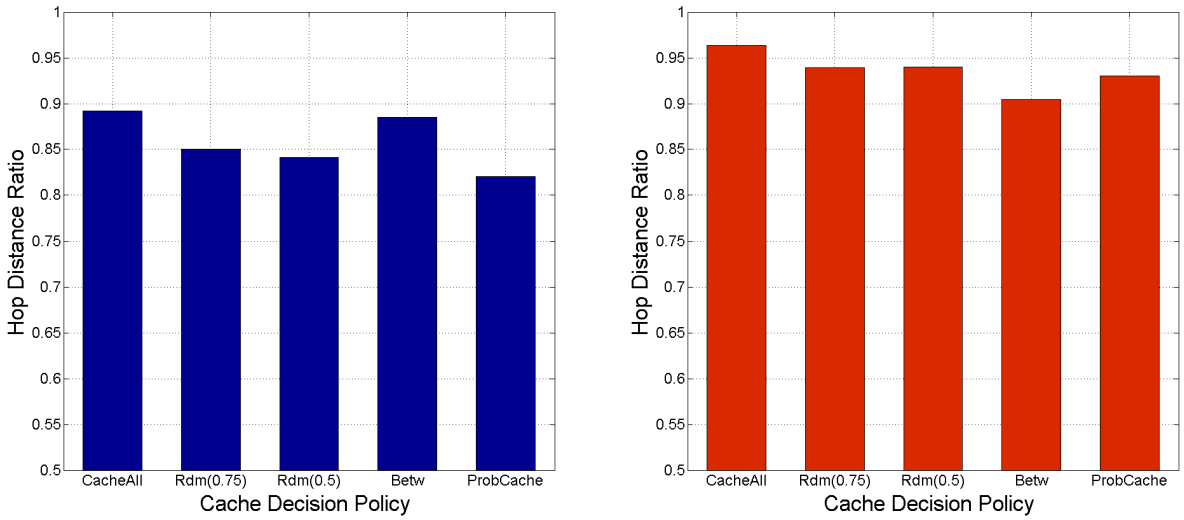
(a) Gateway router

(b) Regional router

(c) Edge router

Figure 3.13: Hit ratio and Content Diversity results under various cache decision policies.

As for regional router, we can observe that the amount of first range contents becomes larger especially in *Betw* and *ProbCache*, in which the sum of three ranges contents climb to around 20% of the cache space. The number of first range contents increase significantly for every decision policy in edge router, in which contents in top three popularity ranges occupy over 20% of content space. *ProbCache* outperforms the others in edge router as the popular contents it cached over 30%. Interestingly, by mapping the hit ratio and Content Diversity results, we find that the proportion of popular contents in the cache is positively related to the Hit Ratio for



(a) Overall request

(b) Miss request

Figure 3.14: Hop distance ratio of decision policy

each decision policy among three types of routers. This result confirms that content popularity plays the most vital role to determine caching efficiency of hit ratio. In addition, we realize that the unpopular contents (contents that not included in top three ranges) are the majority in the caches. Even in the best case of *ProbCache* in edge router, the proportion of unpopular contents is up to 70%. This phenomenon implies the inefficiency of these cache policies in caching the most popular contents.

Although Hit Ratio and Content Diversity metrics offer a view of caching performance at router level, there is no clear sign of how the decision policy behaves at entire network level. And for most of the users, they only concern how long they need to wait before they can reach the contents. To this end, we collect the hop number of contents that travel from content server (or intermediate router if request is satisfied) to the routers that end users connected. We plot the average Hit Distance Ratio for overall and miss contents respectively in Fig.3.14. Hit Distance Ratio equals 0 means all the requests are satisfied at connected router, while 1 means that all the requests are forwarded to content server. Firstly, we inspect the hop reductio ratio for overall requests as labeled in red dash line in Fig.3.14. As we can see, over 10% of traffic are saved by applying in-network caching. In particular, *ProbCache* policy shows its effectiveness of hops saving as it drops the hop distance ratio to around 0.82. On the contrary, hop distance ratio of *CacheAll* and *Betw* is around 0.9, which is much higher than the other three decision policies. Rather surprisingly, the trend of hop distance ratio for miss requests is similar to overall requests except the *ProbCache* policy. The hop distance ratio of *ProbCache*

for miss contents reaches 0.93, which is close to *Rdm* but higher than *Betw*. Different with the other cache policies, *Betw* provides better performance for miss requests than overall requests. This is because *Betw* only enable the cache capability in a subset of routers which are located at higher layer of the network. For most of the requests, they have to be forwarded to cache enabled router to find the contents. Again, *CacheAll* provides the worst caching performance for miss requests as its Hit Distance Ratio is close to 1, indicating that most requests that cannot be satisfied locally are also difficult to find the contents in the intermediate routers along the routing path. Due to the fact that contents are cached in every router along the path, *CacheAll* introduces significant pollution problem to the network. From results of three metrics we discussed above, we summarize our findings as below.

Remark 4 *Indiscriminately caching strategy of CacheAll policy results in high replacement error when LFU is applied and cache pollution in nodes along the delivery path in both single router and entire network level. Caching on a subset of nodes cannot fully exploit the in-network caching in CCN. More importantly, cache decision policy should focus on what contents to be cached rather than where to cache.*

3.5 Principles for Cache Policy Design in Content-Centric Network

Through extensive simulations conducted in our Mobile CCN Caching Simulator (*MCsim*), we evaluate the in-network caching efficiency with multiple parameters in Content-Centric Network (CCN). The results obtained in the simulations provide a higher level view on inspecting the caching issue in CCN. Also, the findings in the simulations can be used for guiding our cache policy design as they provide starting point that we need to deal with. Summarizing our main results, we gather the interesting findings and corresponding design principles as below.

First of all, we find out that the parameter α in Zipf-like distribution is the most significant factor that affects the caching performance in CCN. α parameter can be viewed as the content popularity since it determines how frequently one content is requested. In other words, caching one of the most popular contents is much more efficient than caching hundreds of unpopular contents in both caching performance and cost perspectives. Considering that current Internet traffic follows Zipf-like distribution, cache policy should be able to cache the highly demand contents so that to satisfy the majority of requests. Hence, we give the first design principle:

Design Principle 1 *Cache decision should utilize the property of α parameter in Zipf-like distribution to cache popular contents.*

Secondly, by reviewing Content Diversity results, we realize that the contents we should cached are not always refer to the most popular contents among the entire catalog space. Cache decision making also dependents on where and how the cache routers are organized and located. Thus, hierarchical caching is a more efficient way for improving caching performance for entire network. Interestingly, we find out that cache size is not as important as what we expect. As we can see in the Hit Ratio results, doubling the cache size only provide limited improvement on caching performance.

Design Principle 2 *Cache decision making should dependent on the incoming request traffic pattern rather than arrival contents.*

Thirdly, results of multiple topologies show that routers located at the same routing path suffer serious caching pollution when *CacheAll* policy is applied. In addition, the edge router with direct end users contact outperforms router that only handle forwarded requests. Again, the results confirm that hierarchical caching in the network is necessary as it can avoid the duplicated contents in the same routing path.

Design Principle 3 *Cache decision should achieves hierarchical caching among the network to avoid cache pollution problem.*

Finally, by inspecting the caching performance under existing caching policies, we find that the caching performance improvements provided by existing policies are very limited, especially at low α value situation. Neither topology-assistant ($Betw$) or cache capability based ($ProbCache$) policy can fully exploit in-network caching. Instead, topology-assistant policy might result in traffic congestion and QoS degradation because requests have to travel through the network. Since CCNs enable the automatic caching for every nodes, targeting the contents which can satisfy most requests is more critical than finding the caching location that cache decision policy should consider. In fact, stabilizing the cache is essential for keeping popular contents in the router when LFU replacement policy is applied: caching fewer contents avoids popular contents to be evicted and therefore the access frequency record can remain intact. Meanwhile,

Design Principle 4 *Maintain the cache stability can reduce the replacement error. More importantly, cache decision policy should focus on what contents to be cached rather than where to cache.*

Chapter 4

CachePop: A Popularity Oriented Cache Policy in Content-Centric Network

In this chapter, we design a popularity oriented caching design policy by following the design rules summarized in Chap. 3. Through the simulation results obtained in Chap. 3, we realize that content popularity is the primary factor that affects the in-network caching performance in CCN, especially for today's Internet in which request traffics follow Zipf's like distribution. However, there is no popularity-oriented caching decision policy designed for CCN environment so far. Thus, we propose an efficient popularity oriented caching decision policy: *CachePop* for handling the on-demand contents in CCN.

The organization of this chapter is as follows. Sec. 4.1 briefly introduces several related works on cache policy design in CCN and the problems of these existing decision policies. Sec. 4.2 describes the details of CachePop algorithm and how it is implemented in CCN. We discuss the result table control and X_m value selection in Sec. 4.3. Simulation results is demonstrated in Sec. 4.4 and Sec. 4.5 summarizes this chapter.

4.1 Caching Efficiency Problems of Existing Policies

By evaluating the results obtained in the simulation based study in Chap. 3, we realize that cache decision policy plays an critical role in CCN. This is because decision policy works as a filter to control the content composition in the cache. More importantly, the results show that *CacheAll* policy of CCN is not efficient for caching performance since it cannot differentiate the most valuable (popular) contents. This results in i) the popular contents are easily to be replaced (in low CCR) by unpopular contents (low α value); ii) duplicated contents are cached on the

same route to the server which decrease the overall network caching performance. Although some decision policies such as $Betw$ and $ProbCache$ are proposed to make the caching decision more intelligent. The improvements provided by these policies are very limited because all these decision policies violate the Design Principles; their decisions are made based on router location or topology information instead of the properties of the requests traffic.

4.2 Popularity-oriented Cache Policy: CachePop

In this paper, we design a content popularity-oriented cache decision policy $CachePop$ for CCN by following the design principles we summarized in Chapter 3. Different with previous decision policies, $CachePop$ makes the caching decision based on the incoming requests rather than the contents or the network topology. The advantage of such request-oriented design is that it only targets the highly demanded contents. As we can see in previous simulation results. the request traffics vary according to the network layer that routers located. For instance, regional routers receive more unpopular content requests comparing with edger routers. By moving the decision making to the incoming requests, $CachePop$ can intelligently differentiate and cache the contents that can satisfy most of the incoming request. In another word, $CachePop$ can achieve hierarchical content caching among the network without addition network topology information. Therefore, it can be directly applied on any network topology and still remain the high caching performance.

In order to collect the incoming requests information, we add a Request Table (RT) in CCN architecture. Every time the arrival request cannot find the contents in the router cache, it updates the RT in that router to record accumulated number of requests for the content. When the content is routed back, router checks the RT to determine whether to cache this content or not. We apply a probabilistic function which computes a caching probability for the requested content object and only the content with higher request rate will be cached. The probabilistic function:

$$P(X) = \frac{1}{1 + \beta \frac{X_m^{2\beta}}{X^{\beta+1}}}, \quad \beta = 1.5 \ , \ X = 1, 2, 3, 4... \tag{4.1}$$

where X is the accumulated number of request for the content recorded in the RT. X_m is the parameter that can be tuned according to traffic condition. Since the catalog size in the network is extremely large, the size of RT will increase rapidly, leading to significant processing delay for searching the RT. Thus, RT must remain small and efficient to store the contents that are more likely to be cached later. We apply two methods to control the RT size. Firstly, if one content is cached according to $CachePop$ in the router, the RT will be sorted based on the accumulated number of requests and all the items with the rank larger than cache size will be

Algorithm 1 Pseudo-code of *CachePop* decision policy

Input: Arrival Interest (I) for Data (D)

```
 1: if D is cached in CS then
 2:    copy and deliver D to output
 3: else
 4:    forward I to content server
 5:    search (add) the name and update request number (X) of D in RT
 6:    if RT size > five times of cache size then
 7:       sort items according to X in RT
 8:       remove items with the smallest X
 9:    end if
10:    wait for D
11:    while D arrives do
12:       get the request number X of D in RT and apply probabilistic function P(X)
13:       if cache decision is true then
14:          cache D in content store(apply replacement policy if necessary)
15:          sort items according to X in RT
16:          remove items with the rank larger than cache size
17:       end if
18:       copy and deliver D to output
19:    end while
20: end if
```

Output: forward D to end user

deleted. With this mechanism, we keep the RT size slightly larger than the cache size. Secondly, if the size of RT increases to over 5 times of cache size before one content is cached, the RT will be updated and the all the items with the smallest accumulated number of request are removed.

The Algorithm 1 describes how *CachePop* work in CCN router. When an Interest (I) for Data (D) arrives at router, the router will check its Content Store (CS) to see whether it caches the D that can satisfy I. Lines 1 to 3 is conducted in the cache hit scenario. If D doesn't exist in CS, router forwards I to its neighbors heading to the content server. Meanwhile, it updates the request number for D in Request Table(RT) as shown in line 6. Lines 7 to 10 are used to control the RT size at most 5 times of cache size. And lines after 11 demonstrate the action when D is routed back. After receiving D, the router applies probabilistic function $P(X)$ to determine whether to cache the arrival Data D or not rather than cache all the Data indiscriminately. If cache decision is true, router will add the D into its CS and evict Data if necessary (line 14 to 15). Lines 16 to 17 describe another mechanism for RT size control which limits the RT size to the same order as cache size. At the end, the D will be forwarded to the end user.

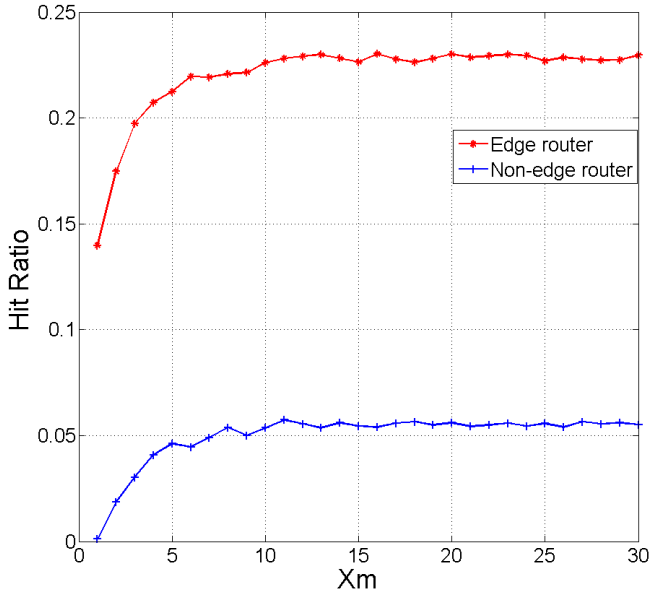


Figure 4.1: Hit ratio under Xm setting

4.3 Request Table Control and X_m Selection

In our *CachePop* algorithm, we add the mechanism to control the size of Request Table (RT). By applying this mechanism, RT will remove items with the smallest accumulated request number when its size reaches the threshold, keeping the RT size as the same order as cache size. This mechanism is necessary for two reasons. First of all, the size of RT is expanded rapidly as all the requests are recorded in RT, leading to significant processing delay for *CachePop* decision policy. To avoid violating the line-of-speed requirement in CCN, the size of RT must keep in small scale. In addition, we know that the request traffic follows Zips like distribution, which means that most of the contents are requested infrequently. Therefore, recording the request information of less accessed contents is not necessary since $P(X)$ function only allow contents with high request number to be cached.

Now we consider the selection of parameter X_m in probabilistic function $P(X)$. In fact, X_m controls the tightness of *CachePop* decision policy. Small X_m increases the probability that contents with small request number can be cached. On the contrary, only contents with high access frequency are likely to be cached when X_m is large. In order to find the optimal X_m for achieving the best caching performance, we conduct the simulations in line topology applying *CachePop* decision policy. The settings of simulation are the same as before with α value =0.8 and CCR=1%. We increase the X_m value from 1 to 30 and the result is shown in Fig. 4.1. From

the result we can observe that for both edge and non-edge router, the hit ratio increases rapidly at the beginning where $X_m \in [1, 5]$. The increasing rate of hit ratio slows down at $X_m \in [5, 10]$, and after $X_m > 10$, the hit ratio reaches maximum values at 0.23 and 0.06 for edge and non-edge router respectively. In $P(X)$ function, X_m works like a valve to controls the difficulty that one content to be cached. After reaching the threshold value (10 in our simulation), the caching performance is close to the best under the α and CCR setting. Therefore, the improvement of hit ratio after X_m larger than 10 is negligible.

Notice that though larger X_m can optimize the hit ratio performance, it also slows the adaptation of *CachePop* for handling the α (content popularity)change in request traffic. In addition, the pending table in CCN actually distorts the request traffic pattern since only one request is forwarded for the content. Thus, we select X_m to be 10 which is the optimal value when considering caching performance and adaptivity.

4.4 Simulation Results and Discussion

4.4.1 System and Scenario Description

Table 4.1: System parameters notation.

Parameter	Meaning	Values
R_d	Request traffic distribution	Zipf
α	Zipf exponent	0.8
R_t	Total request from aggregated users	10^5
R_r	Request rate	25Hz
P	Packet size	1
c	Cache size	100
C	Catalog size	10^4
$\frac{c}{C}$	Cache/Catalog ratio	1%
CDP	Cache decision policy	*CacheAll*, *Rdm*(0.5), *Rdm*(0.75) *Betw*, *ProbCache*, *CachePop*
CRP	Cache replacement policy	*LRU*
T	Network topology	Hybrid

According to the results shown in Chap. 3, CCN caching performance suffers serious cache pollution and replacement errors at low α and CCR environment. In addition, the effects of cache policies on caching performance are minimized when α value becomes large because large α value is the dominate factor that controls the popularity of contents in request traffic. In order to better evaluate our proposed cache policies, the simulation environment should be set

in the case which similar with the real world. Thus, we use the default settings with α =0.8 and CCR=1% as we analyze in Table. 3.4. We assume that there is no packet loss.

4.4.2 Performance Metrics

For caching performance evaluation of on-demand contents, we use the same performance metrics as in Chap. 3 and they are *Cache Hit Ratio, Content Diversity* and *Hop Distance Ratio.*

I. Cache Hit Ratio

Cache Hit Ratio describes the percentage of requested contents can be found in the cache. The definition of *Cache Hit Ratio* is shown in Eq. (3.2), which varies between the interval [0,1] with the lower bound achieved in the worst case that no request is satisfied at the cache. *Cache Hit Ratio* is simple but useful as it straightforwardly reflect the utilization of cached contents

II. Content Diversity

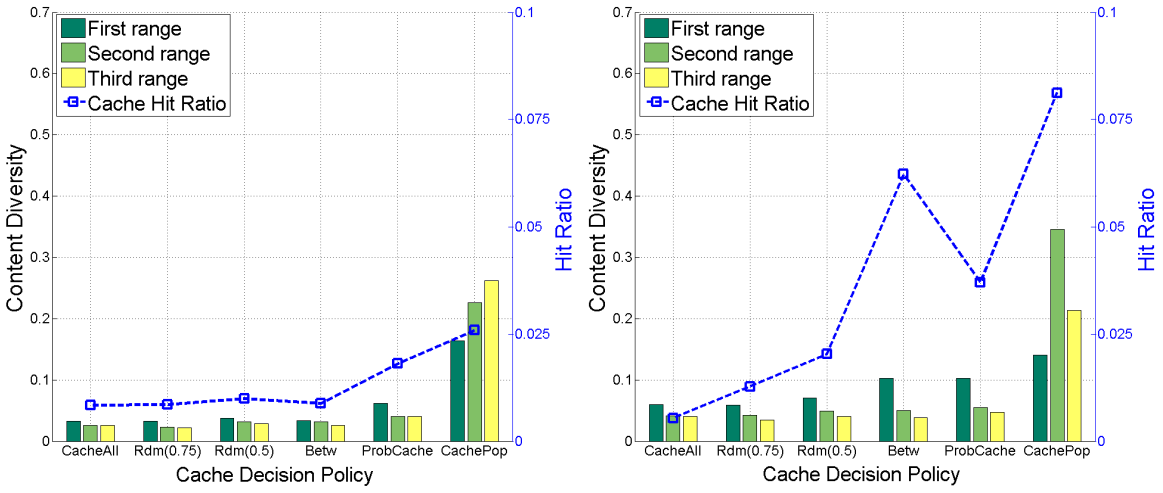
Content Diversity defined in Eq. 3.3 demonstrates the composition of cache in routers. As a popularity-oriented cache policy, *CachePop* should have the ability to differentiate arrival contents based on their popularity and cache the most popular ones. And as we discussed in Sec. 3.4.2 and Sec.3.4.4, hierarchical content caching is a more efficient way for improving the caching performance of the entire network. Thus, *Content Diversity* is the suitable and necessary metrics for our caching performance evaluation.

III. Hop Distance Ratio

In users perspective, the latency of content retrieval is the most important factor that impact the QoS experience. In our simulation, we evaluate the content retrieval latency by hop counts. Intuitively, requested contents should be cached close to the end users so that they can be fetched faster than from the content server. Lower hop distance ratio means less bandwidth consumption and lower content delivery latency that are desired by network operator and individual user respectively. We define user-centric metric *Hop Distance Ratio* in Eq. (3.4), which describes the average hop distance that one request can reach its content.

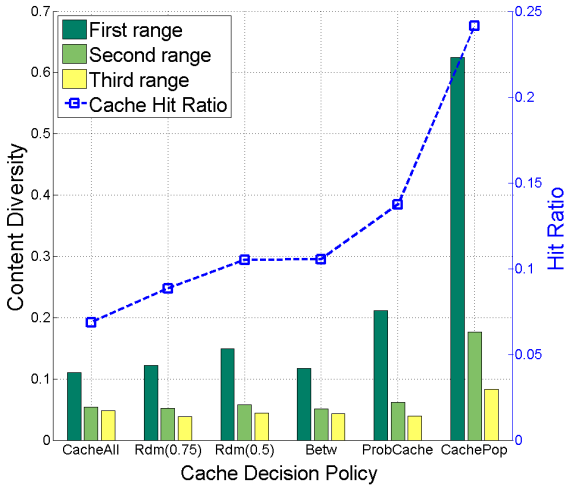
4.4.3 Results of *CachePop* Decision Policy

We apply *CachePop* decision policy in all 13 routers via $MCsim$. This simulation in conducted in hybrid topology with the same environment settings used in our simulation study in Sec.3.4.5. We compare the caching performance of *CachePop* against existing cache decision policies that evaluated in Sec.3.4.5.



(a) Gateway router

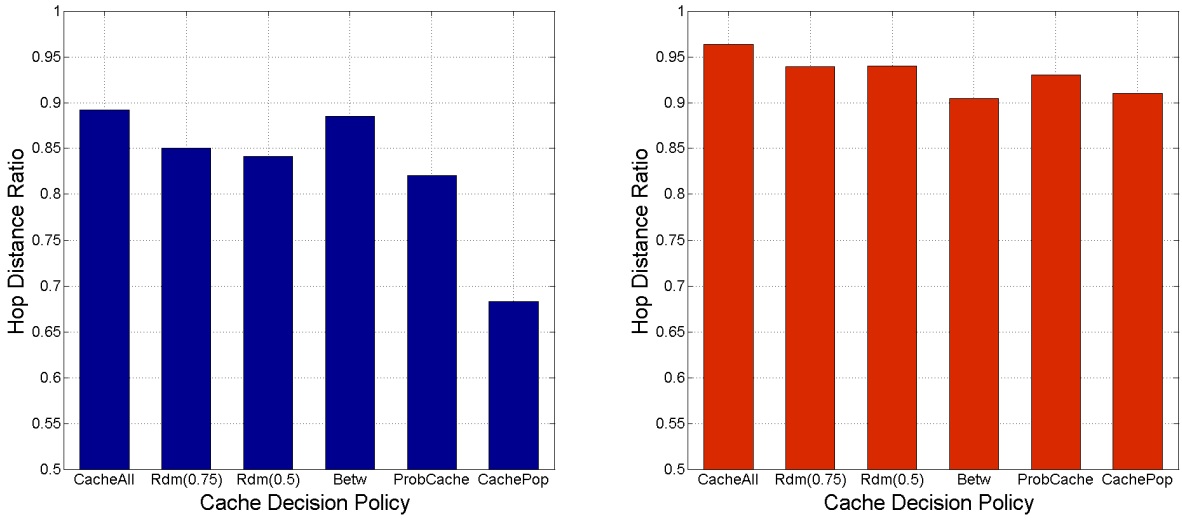
(b) Regional router



(c) Edge router

Figure 4.2: Performance results of *CachePop* for on-demand content.

Fig. 4.2 shows the comparison of *Cache Hit Ratio* and *Content Diversity* results among 6 different decision policies. We collect the average results at three types of routers accordingly. As we observe that, the hit ratio values of *CachePop* performs the best among all three types of routers. Especially at edge router (shown in Fig. 4.2c), the hit ratio of *CachePop* reaches 0.24, which is around 71% improvement comparing with the second best hit ratio result from *ProbCache*. And it is over 240% improvement comparing with widely adopted *CacheAll* policy. According to Zipf-like distribution, the maximum hit ratio at edge router can be achieved only



(a) Overall request

(b) Miss request

Figure 4.3: Hop distance ratio of decision policy

if 1) all the first range contents are cached and never be replaced in the router 2) requests are sent directly from end users. And the theoretical maximum hit ratio under our simulation setting is 0.3. However, avoiding intermediate routers among the network is costly and also inefficient for content delivery. Thus, this theoretical maximum hit ratio cannot be achieved in reality and we argue that the 6% performance difference is relatively small in small α and CCR value environment when intermediate requests and cache replacement error are taken in consideration. Similar trend can be observed in both regional router (Fig. 4.2b) and gateway router (Fig. 4.2a) as popularity-oriented *CachePop* outperforms the other decision policies. Due to the special caching design, *Betw* performs close to *CachePop* at regional routers as they have highest betweenness value. Notice that at routers located at higher level in the network, the *Cache Hit Ratio* are more difficult to be improved.

By investigating content diversity in Fig. 4.2, we can see the dramatically difference of content composition in cache between *CachePop* and the rest decision policies evaluated in the simulation. For example, in edge router results shown in Fig. 4.2c, over 60% cache space are used to cache first range (top 100 most popular) contents. However, in regional router case (shown in Fig. 4.2b), we can see that second range contents are the majority (34%) in the cache instead of first range contents. This is because that most of the first range contents can be reached at the edge routers and therefore, more requests for less popular content are forwarded to regional routers. The similar result can be found in gateway router; by making the cache decision based on the arrival requests, *CachePop* cache more contents(25%) from third range . Notice that it is

more harder to cache high popular contents at upper layers since the arrival request is modified by lower layer routers and also the PIT effect which avoids same requests to be forwarded. Thus, *CachePop* is extremely effective to improve the caching performance for on-demand contents as it successfully identify the most valuable contents and achieve hierarchical caching among the network.

Hop Distance Ratio results are shown in Fig. 4.3. As we can observe that, for overall requests, the average hop distance ratio of *CachePop* drops down below 0.7 which is about 15% smaller comparing with the performance of *ProbCache* (0.83) and the rest decision policies with the *Hop Distance Ratio* vary between 0.85 to 0.90. This result confirms that by applying *CachePop*, requested contents can be accessed in a shorter distance and over 30% of bandwidth are saved in CCN compared to the network architecture without using in-network caching. We also find that the *Hop Distance Ratio* of overall requests in *Betw* is higher than any other policies except *CacheAll*. This is because *Betw* only allows router with high betweenness value (regional routers in hybrid topology) to cache the contents, leading to traffic congestion on higher level of the network. In network operator's view, *Betw* is more efficient since it aggregate the requests traffic from the lower level of the network. However, as shown in the figure, it is not suitable for end users because it increase the overall delay. We also plot the *Hop Distance Ratio* for miss requests that cannot be satisfied at edge routers. The hop distance ratio for miss request in *CachePop* is around 0.9 which is close to *Betw* and smaller than the rest. Considering that the short distance from edger to the gateway router (only 2 in hybrid topology) and less cache space in non-edge routers(4 non-edge routers vs. 9 edge routers), we argue that any small improvement in hop distance ratio for miss requests is considerable as it demonstrates more efficient utilization of *CachePop* to achieve hierarchical content caching in the network.

So far, the simulations we conducted use fixed $\alpha = 0.8$. However, the content popularity in real Internet request is changing over time. Different with other existing decision policies, *CachePop* makes the caching decision based on the request number recorded in the Request Table (RT). Therefore, we want to examine whether such mechanism introduces decision latency that impacts the caching performance. We conduct the simulation with CCR=1% and α varies from 0.6 to 1.2. Each router receives 10^4 requests from end users at every α value. In order to warm up the cache at the very beginning, each router receives additional 10^4 requests with the content popularity factor α =0.6. Thus, the total requests received by each router is 5×10^4. *Cache Hit Ratio* is collected every 100 request at each edge router and we get the average value of all edge routers as shown in Fig. 4.4.

The result starts from the $10^4 th$ requests after the cache warm up and the *Cache Hit Ratio* is stable in the first 10^4 requests for all decision policies. The *Cache Hit Ratio* of *CachePop* reaches 0.11 at α =0.6, which is much higher than the other policies. After the request number goes beyond 2×10^4, the α is increased to 0.8 and fixed at 1.2 at the end. Clearly, the *Cache*

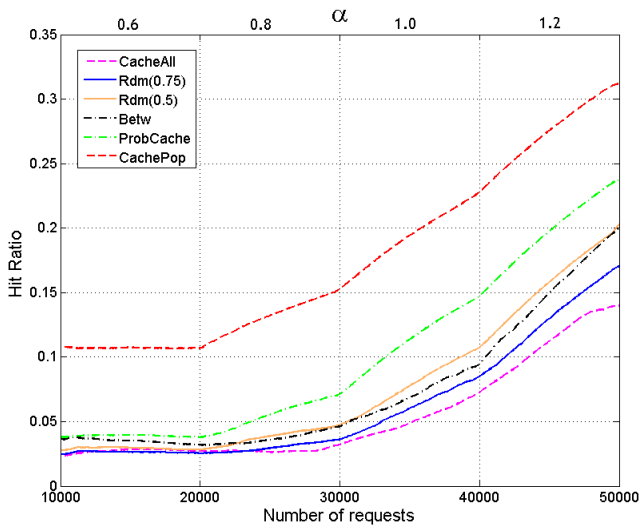


Figure 4.4: Responding rate to request traffic change

Hit Ratio of different decision policies increase with various increasing rate after the α value changes. For *CacheAll* and $Rdm(0.75)$, the increasing speed is slower against the others at α =0.8. While we can observe that increasing rate of hit ratio for *CachePop* and *ProbCache* changed immediately each time after the α value changed and their increasing speed are the fastest among all the decision policies. This result demonstrates the fact the *CachePop* not only significantly improves the caching performance but also quickly responds to content popularity change.

4.5 Summary

In this chapter, we follow the Design Principles summarized in Chap. 3 and propose the popularity-oriented caching decision policy (*CachePop*). In particular, we add a Request Table in CCN architecture for recording the arrival requests information. We specify the optimal Xm valued used in decision making process and Request Table control method. We validate our *CachePop* cache policy through *MCsim* with respect to Cache Hit Ratio, Content Diversity and Hop Distance Ratio. The simulation results show that *CachePop* significantly improves the caching performance in both the node and network level compared to the other policies. By inspecting the content diversity results, we can observe that *CachePop* successfully identifies and cache the most popular contents in arrival requests and achieves hierarchical caching. In addition, simulation result demonstrates that *CachePop* can quickly responds to content popularity change in request traffic.

Chapter 5

NG-CachePop: A Neighbor Graph Based CachePop in Mobile Content-Centric Network

In this chapter, we propose the mobility supported popularity oriented cache policy by optimizing *CachePop* designed in Chap. 4. In order to achieve our goal of improving both on-demand content and real-time content delivery, we break our cache policy design into two steps in this research. The first step is to design an efficient cache policy to improve the caching efficiency for on-demand contents which are the majority of current request traffic. We complete step one in Chap. 4, in which a popularity-based cache policy *CachePop* is proposed to meet the requirement. However, popularity-oriented caching scheme will exacerbate the handover latency (retransmission delay) of real-time contents which will not be cached since most of them are unique (unpopular) and only be requested for individual users. Therefore, the second step of our design is to modify the *CachePop* cache policy so that it can reduce the retransmission delay for real-time contents after the handover. We propose our final cache policy design called *NG-CachePop* by applying an innovative neighbor graphs and proactively caching feature on *CachePop*. As a mobility supported popularity oriented cache policy, *NG-CachePop* guarantees the caching performance for on-demand content but also achieve the seamless handover for real-time contents simultaneously.

The organization of this chapter is as follows. Sec. 5.1 reviews and analyzes the handover latency problems introduced by existing caching decision policy. Sec. 5.2 to Sec. 5.4 describe the implementation of neighbor graph, proactive caching routing mechanism and operations details respectively. At the end, we evaluate the performance of *NG-CachePop* in our simulation platform *MCsim* in Sec. 5.5.

5.1 Handover Latency Problem in CCN

Mobility support is one of the beneficial properties introduced by CCN as we discussed in Sec. 1.2. In contrast to TCP/IP's connection-oriented end to end control, CCN deliver packets based on the name of the content in hop by hop manner. With this location/identity split principle in CCN architecture, mobile user can directly re-send the *Interest* packet for contents that have not been received before the handover process. Such mobility support scheme is more robust as it doesn't require the connection re-establishments that relevant to the locations of two ends. Therefore, CCN doesn't involve the triangular routing problem and large control overhead issue that existed in current Mobile IP network.

Despite the robustness and efficiency of mobility support in CCN, the increasing handover latency is unavoidable when mobile user sends an *Interest* packet for content and moves to a new attachment point before the content is received. In such scenario, the delay for receiving the content can up to two times of RTT plus attachment points connection delay. This delay may still meet the requirement of on-demand contents, but it would impact the user-experience for real-time applications with strict delay requirement such as multimedia streaming, stock trading and online gaming. In addition, different caching schemes utilized in CCN can affects the delay for real-time application as well. For example, the performance of $CacheAll$ policy is not as good as $CachePop$ policy in our simulation result in Chap. 3 when only on-demand contents are requested in the network, while it may not be the case when real-time contents are involved since 'unique' real-time contents will not be cached by $CachePop$ in the cache on the route.

In order to evaluate the how caching schemes affect the handover latency, we conduct a simulation via our $MCsim$ in mobile CCN scenario in which a mobile user sends requests (request rate = 5Hz) for real-time contents during the movement. When the Interest packet is sent by the user, it will be inserted into the pending list and only the arrival contents can consume items in the list. If handover happens, mobile user will re-send all the *Interests* that have not been satisfied in pending list after connecting to the new point of attachment(router). The goal of this simulation is to evaluate the performance of existing caching schemes ($CachePop$ included) at mobile scenario especially the period right after the handover process. For simulation settings, we apply hybrid topology in a $3km * 3km$ area. All nine edge routers are uniformly located in this area. The α value of on-demand traffic request is 0.8 and the CCR value is 1%. We assume that real-time contents are unique and the real-time traffic request follows poisson distribution. In this simulation, mobile user moves according to random waypoint model. We apply LRU as the cache replacement policy and evaluate $CacheAll$, $Rdm(0.5)$, $ProbCache$ and $CachePop$ decision policies in such mobile scenario.

In normal case when mobile user is moving within the area of the same routers, the caching

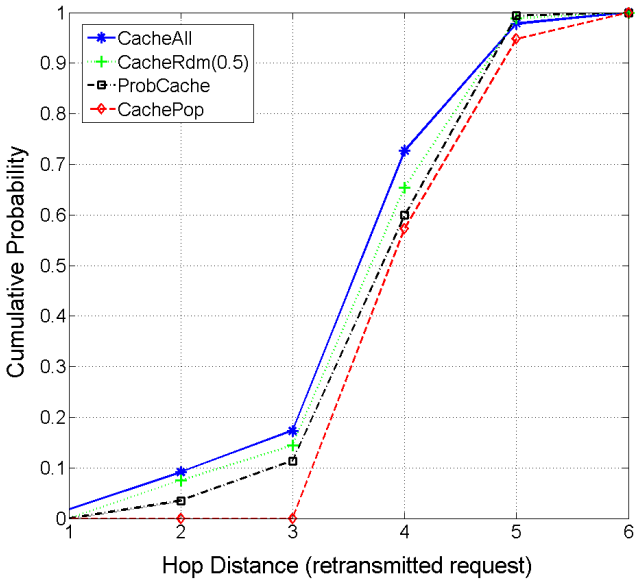


Figure 5.1: Number of hops for retransmitted Interest to reach requested content

performance is the same as static user when signal strength and landscape is not considered. Therefore, for results analysis, we only consider the retransmitted contents since our point of this simulation is to evaluate the delay for contents that received right after handover. We collect the number of hop that retransmitted contents travel through the network. Minimum number of hop for retransmitted content is 1 which means that content can be accessed immediately as it is cached at the connected edge router after the handover. Maximum hop number is recorded when the re-transmitted *Interest* cannot be satisfied until reaching the content server. In hybrid topology, the maximum hop number is 4 at edge router 5,6,7,8 and 13, while it is 5 at edge router 9,11,12 and 6 at edge router 10. Intuitively, larger hop number indicates higher delay, which is not acceptable for most of the real time applications.

Fig.5.1 shows the cumulative probability of hop count for retransmitted *Interests*. First of all, we can find that most of the (over 82%) of retransmitted *Interests* cannot be satisfied within 3 hops distance, indicating that these *Interests* are forwarded along the path to the content server. Among all cache decision policies, *CacheAll* performs slightly better than the others as 18% of contents can be accessed within 3 hop distance and about 3% of the contents can be reached immediately after handover. The performance of $Rdm(0.5)$ and *ProbCache* are similar; about 10% of contents can be found within 3 hop distance. We also observe that, different with *CacheAll*, no retransmitted contents can be accessed in 1 hop distance when $Rdm(0.5)$, *ProbCache* or *CachePop* decision policies is used. In fact, *CachePop* introduces

significant delay for content retrieval right after the handover since no retransmitted *Interests* can be satisfied within 3 hops. This is because that *CachePop* does not cache any real-time contents as they are considered unpopular for the other end users. Although *CacheAll* policy caches every arrival real-time contents, it is difficult to keep them in the router since these real-time contents are replaced by later arrival contents in a short time interval.

By analysing previous simulation result, we realize that the handover latency for real-time contents are significant since most of these contents cannot be found within the network. This reactive mobility support provided by CCN cannot meet the delay requirement for most of the real-time applications. In addition, popularity-based *CachePop* decision policy that designed for handling on-demand contents deteriorates the contents retrieve performance when real-time contents are involved. Based on the observations in the result, we summary two aspects that need to be considered to improve the *CachePop* decision policy for both on-demand and real-time applications in mobile scenario.

- Different cache decision schemes should be applied based on the type of the contents. As proved in Fig. 5.1, popularity-based caching scheme greatly improves the caching performance for on-demand contents delivery. However, *CacheAll* policy is more suitable for real-time contents since they can be cached within the network and more likely to be found after mobile's handover.

- Proactive caching is required to decrease the hop distance for retransmitted *Interests*. In order to minimize the delay, the hop count for retransmitted *Interests* should be narrowed down to 1 (seamless handover) so that the retransmitted *Interests* can be satisfied immediately at the connected edge router after handover.

5.2 Neighbor Graphs

In previous section, we already demonstrated that real-time contents suffer significant handover latency when mobile user moves from old PoA to new PoA. The results confirm that reactive mobility support of CCN architecture is not sufficient to satisfy real-time applications which have strict time requirement especially when popularity-based caching schemes are utilized. Hence, a proactive caching solution is necessary to improve the handover latency for real-time applications. This idea can be achieved when the requested contents can be proactively cached in the "neighbor" routers within one hop distance. Notice that "neighbor" routers here do not refer to physically connected routers but routers that mobile user connected to after handover phase. According to the routers deployment and network organization, neighbor routers can be located far away from each other or even in different sub-networks.

In all, we are looking for a method that can provide an appropriate candidate set of router, in which requested contents are cached before the retransmitted *Interest* packets arrive. We apply Neighbor Graph (NG) as it can identify such candidate router set in a light-weight manner without knowing mobile's movement. NG has been proposed for fast handoffs in IEEE 802.11 (Wi-Fi) network [89, 90] and WLAN [91]. The motivation of applying NG is that, by receiving topological information, the router generates neighbor graph which contains set of one-hop-distance routers that mobile user is heading to after the disconnection. It is likely that the concept of NG can be directly applied to CCN architecture since the mobile user is moving to one of the one-hop-distance routers which contain the contents that have not been received before the handover phase. However, the application of such an idea in CCN is quiet different compared with traditional TCP/IP networks. In [89–91], the goal of applying NG is to proactively send the contents to neighboring routers according to the address, whereas in CCN the objective of applying NG is to forward the content to satisfied the *Interest* inserted in the pending list of routers along the path. Due to the nature architecture difference, more specific issues about *Interest* packets forwarding should be taken into consideration. We introduce the details of applying NG in CCN later.

5.2.1 Definition of Neighbor Graphs

Before introducing our proposed NG based mobile caching scheme, we briefly review the concept of NG and how it is generated. NG is an undirected graph which captures the *one-hop-distance relationship* between point of attachments (PoA). We consider routers as the PoA in our research. As mentioned earlier, *one-hop-distance relationship* refers to the case that mobile user reaches the coverage limits of current router R_i and connects to new router R_j after the handover procedure. *one-hop-distance relationship* dependents on many factors such as router deployment, landscape and signal strength. In most of the cases, *one-hop-distance relationship* is directly related to the physical distance or mobility path between two routers.

Fig.5.2 shows an example of neighbor graph with 5 routers. Routers R1-R5 are located in the area and the dashed line represents the possible movement traces of mobile user. At the beginning, we assume that the mobile user is moving away from R1. As we can see, there are only two potential pathes with the destinations to be either R2 or R4. This means that if mobile user keeps moving, handover procedure will occur and the mobile user will connect to either R2 or R4. Notice that R1 and R5 is not considered for *one-hop-distance relationship* because mobile user has to connect to either R2 or R4 before reaching R5. The same result applied to R3 where no router has *one-hop-distance relationship* with it except for R4. After going through all possible paths existed in physical topology, we can the convert the topology information into the neighbor graph as shown on the right side of Fig. 5.2. Notice that although

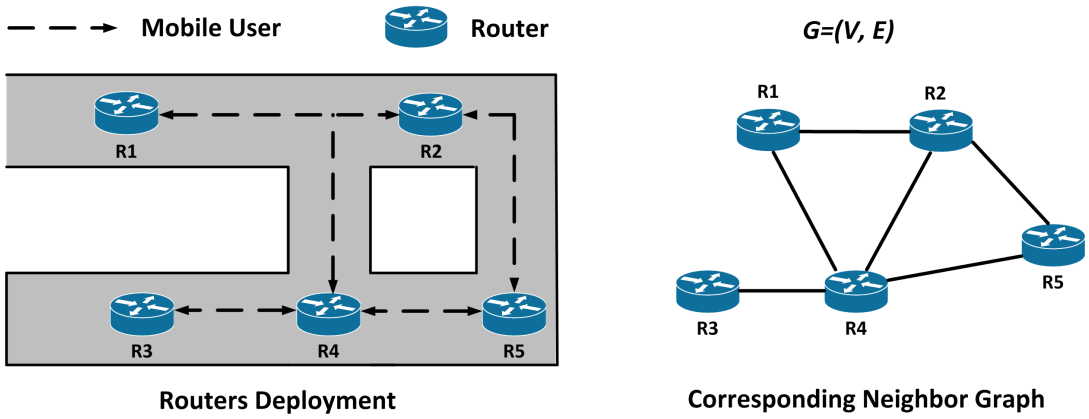


Figure 5.2: Example of neighbor graph with 5 routers

R3 and R4 are located with the same distance to R1 in physical topology, there is no direct edge between R1 and R3, indicating that short physical distance cannot guarantee the *one-hop-distance relationship*.

We define a undirected graph as

$$\begin{aligned} &G = (V, E) \\ &V = R_1, R_2, ...R_n \\ &E = (R_i, R_j) \\ &N(R_i) = \{R_k : R_k \in V, (R_i, R_k) \in E\} \end{aligned} \tag{5.1}$$

where G is the data structure of neighbor graph; V is the set for all routers; e is the edge between R_i and R_j if they have *one-hop-distance relationship* and $N(R_i)$ is the set for all the neighbor routers of R_i.

Generation of Neighbor Graphs

In order to apply neighbor graph, one of the most essential tasks is to convert the physical topology into corresponding neighbor graph as shown in Fig.5.2. In our research, we implement neighbor graph in a distributed manner, through which each router updates and stores its set of neighbors automatically. In CCN scenario, before the handover, mobile user records the name of the old router R_{old} (e.g. $/*/NCSU.edu/maincamp/R_{old}$). After connecting to the new router R_{new}, mobile user generates a special *Reconnection Request* carrying the name of the R_{old} and sends it to newly connected router R_{new}. R_{new} checks its neighbor graph set $N(R_{new})$ after receiving *Reconnection Request*. If the R_{old} already existed in $N(R_{new})$, the packet will be discarded. Otherwise, R_{new} adds R_{old} into its neighbor graph set $N(R_{new})$ and forwards this *Reconnection Request* to R_{old} routed by the name of R_{old}. After receiving *Reconnection*

Algorithm 2 Pseudo-code of neighbor graph generation at router R_i

```
1: if receive Reconnection Request from user then
2:    if R_old existed in the list of neighbors then
3:       update timestamp of R_old
4:    else
5:       delete outdate router in neighbor list in LRU fashion if necessary
6:       add R_old as a neighbor
7:       send Move-Notify to R_old
8:    end if
9: end if
10: if receive Move-Notify from R_new then
11:    if R_new existed in the list of neighbors then
12:       update timestamp of R_new
13:    else
14:       delete outdate router in neighbor list in LRU fashion if necessary
15:       add R_new as a neighbor
16:    end if
17: end if
```

Request, R_{old} realizes that R_{new} is its neighboring router and adds R_{new} in the neighbor graph set $N(R_{old})$. R_{old} will send a *Move-Notify* back to R_{new} to notify that the edge has been added.

The Algorithm 2 describes how neighbor graph is generated at each router. In order to make sure all the routers in neighbor graph set are alive, each router maintains its neighbor graph set in a LRU manner. This means that the size of $N(R_i)$ is fixed and it is set to be 3 or 4 depending on the routers density. $N(R_i)$ control is necessary as it guarantees the freshness of the neighbor graph and avoids overheads due to too much duplicated contents forwarded in the network. In addition, edge that is incorrectly added can be eliminated in a short time as LRU can detect and remove it. According to the Algorithm 2, we can see that the neighbor graph of each router is constructed independently and only the first a few users suffer high handoff delay when neighbor graph is not fully constructed.

5.3 Proactive Caching Scheme Design in CCN

As we discussed earlier, the motivations of using neighbor graph for improving the handover latency are likely the same among different network. However, the implementation of neighbor graph based proactive caching in CCN is not trivial as the routing process in CCN is fundamentally different. By implementing neighbor graph, router R_i in CCN carries a candidate neighboring router set $N(R_i)$ that mobile user may move to. Intuitively, proactive caching allows contents that have not been received by mobile user during the handover procedure to

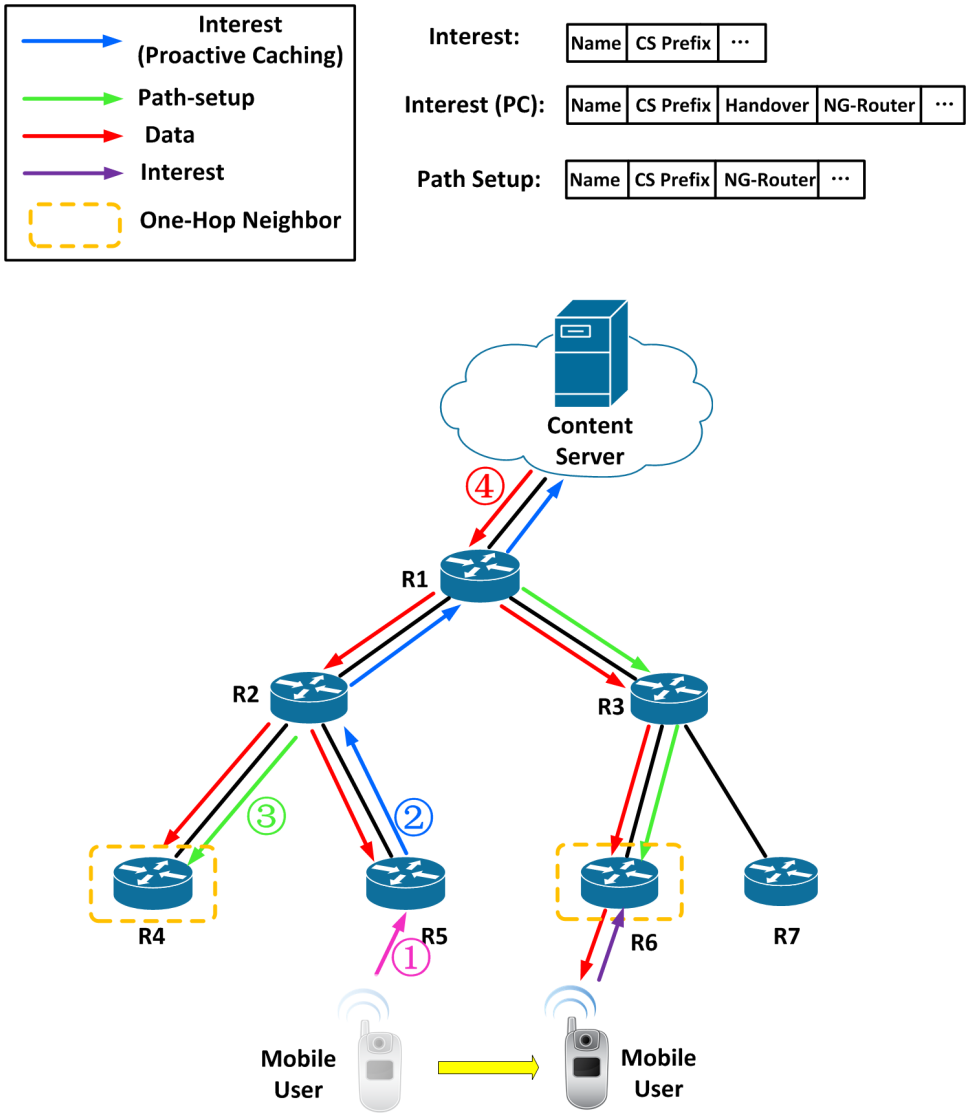


Figure 5.3: Operation of NG-based proactive caching in CCN

be forwarded to neighboring routers. Hence, mobile user can reach the contents directly at the connected router after the handover rather than waiting for the contents delivered from content server. This concept can be easily achieved in traditional IP network via IP multicast as long as IP addresses of the neighboring routers are known. However, it is not the case in CCN. As a receiver-driven model, the conversation between two ends in CCN starts with an *Interest* packet sent by the receiver. Router forwards the *Interest* packet based on the name of requested content and inserts the *Interest* into pending list. After content is found, it is routed back to the user following the reverse path of *Interest* packet. In other word, *Interest* packets in pending

list work like road signs to help content find the path back to end user. In order to redirect the contents to the routers in the neighboring graph set before the arrival of retransmitted *Interest*, we propose the proactive caching method for CCN. The operation of proactive caching method is largely made of three steps with detailed procedures described as follows. An example is shown in Fig.5.3.

Step I. Handover notification: When mobile user (MU) approaching the edge of connected router coverage, it sends an proactive caching required *Interest* packet (*Interest(PC)*) to its old router. *Interest(PC)* (shown in Fig.5.3) has two more fields, $Handover$ and $NGRouter$, compared with original *Interest* packet.The purpose of *Interest(PC)* is to notify receivers that this requested content needs to be proactively cached at routers listed in $NGRouter$. MC sets the $Handover$ flag and leaves $NGRouter$ blank. In Fig. 5.3, MU detects the handover and therefore it sends an *Interest(PC)* to notify router R5 that this requested content should be redirected to its possible next connected router.

Step II. Neighbor graph assistance: Only the router that receiving *Interest(PC)* directly from MU inserts its neighbor graph set (one-hop-distance neighbors) into $NG\ Router$ field of *Interest(PC)*. Every router receiving *Interest(PC)* forwards this packet to Content Server (CS) based on FIB reference and adds the *Interest*(PC) into PIT as dealing with original *Interest* packet. After that, router checks its FIB to see whether it has interface(s) that can reach candidate routers recorded in $NGRouter$ field. If such interface is found and it is different with the in/outgoing interfaces of *Interest(PC)* , router generates $PathSetup$ message (shown in Fig.5.3) and sends it to that candidate router. In Fig.5.3, R5 receives *Interest(PC)* from MU directly. Thus, R5 inserts its neighbor graphs set (R4 and R6 in our example) into $NGRouter$ and forward the *Interest(PC)* to R2. R2 receiving *Interest(PC)* checks its FIB to find the interface for R4 or R6 and only the interface to R4 is qualified. Thus, R2 generates a $PathSetup$ message by adding the R4's prefix on original *Interest* and sends it via the qualified interface. The same as R2, R1 generates $PathSetup$ message heading to R6 because the interface to R6 is different with in/outging interfaces of received *Interest(PC)*.

Step III. Forwarding path setup: When router receives $PathSetup$ message, it checks the $NeighborRouter$ field and determines whether it is the router in that field. If so, relevant PIT entry is created based on the information stored in $PathSetup$ message. The requesting interface is left blank since the *Interest* packet has not arrived yet. If router notices that it is the intermediate router, it forwards the $PathSetup$ message to end router based on the FIB information. After that, it creates the PIT entry in which the requesting interface is the outgoing interface for forwarding the $PathSetup$ rather than ingoing interface as in normal *Interest* forwarding process. This is because that the goal of $PathSetup$ message is to find a path that retransmitted *Interest* may go through but in reversed direction. With this message, on-the-fly contents can be redirected to all the neighbor graph routers that MU may connected

to after handover without breaking the routing in CCN architecture. As shown in Fig.5.3, after receiving *PathSetup* message, R3 creates relevant entry in PIT and forwards the *PathSetup* message to R6 since it is not the neighbor graph routers of R5. Both R4 and R6 just add the entry in their PIT as they are the destinations of requested content.

Step IV. Content delivery: In our design, content forwarding is exactly the same as original CCN architecture design. Arrival contents are forwarded based on the requesting interfaces list in PIT entry. As one of the most promising feature in CCN, PIT allows multiple interfaces data transmission. To this end, content is forwarded to the destinations based on the route set up by both *Interest(PC)* and *PathSetup* messages. As we can see in Fig.5.3, Data packet is forwarded to and cached at R4, R5 and R6 simultaneously. When MU reconnects to R6, it can reach the content that requested before handover with one hop delay only.

5.4 Operation of NG-based CachePop Policy

By applying neighbor graph discussed above, CCN now achieves proactive caching during the handover process as the requested contents can be successfully forwarded to the neighbor routers. However, these requested contents will not be cached when *CachePop* is used. Thus, mobile user with slow mobility will miss the contents if they arrive after the contents being removed. Keep in mind that our goal of cache policy design is to handle caching efficiency and handover latency at the same time. Therefore, we let NG-based *CachePop* makes cache decision based on the type of arrival contents.

Considering that on-demand content follows Zipf-like distribution, NG-based *CachePop* applies original *CachePop* for arrival on-demand contents. On the contrary, NG-based *CachePop* simply applies *CacheAll* for caching real-time content which requires proactive caching for seamless handover. As we discussed in Chap. 3, *CacheAll* decision policy may decreases the caching performance because of cache pollution. If on-demand and real-time content share the same cache space, real-time contents will dominate the cache since the replacement rate of *CachePop* policy is relatively small compared to *CacheAll* policy. In order to avoid the cache pollution result from *CacheAll* policy, NG-based *CachePop* policy splits the router's cache into two parts: regular cache and mobile cache which are responsible for *CachePop* and *CacheAll* respectively. In order to improve both the cache performance and handover latency, the ratio of regular over mobile cache must be carefully controlled.

5.5 Simulation Results and Discussion

5.5.1 System and Scenario Description

Table 5.1: System parameters notation.

Parameter	Meaning	Values
R_d	Request traffic distribution	Zipf, non-duplicated random
α	Zipf exponent	0.8
R_t	Total request from aggregated users	10^5
R_r	Request rate	5Hz, 25Hz
N_m	Number of mobile users	25, 50, 100
P	Packet size	1
c	Cache size	100
C	Catalog size	10^4
$\frac{c}{C}$	Cache/Catalog ratio	1%
CDP	Cache decision policy	$CacheAll$, $Rdm(0.5)$ $CachePop$, $ProbCache$
CRP	Cache replacement policy	LRU
M	Mobility trace	NCSU main campus traces
T	Network topology	Hybrid

In mobile scenario, we map hybrid topology onto NCSU campus ($3000m \times 2400m$) with nine edge routers are deployed uniformly in this area. We use four real GPS traces [92], which reflect the human movements in NCSU campus. The mobile traces are shown in Fig. 5.4, in which we can find that there are some overlapping sessions among four traces. For instance, R9, R10 and R12 have heavy traffic since most of the traces go through these three areas. These trace plots correspond to the human movement preference or road directions in NCSU main campus. During the simulation, each mobile user randomly selects one of the mobile traces and repeats the movement. Different with non-mobile aggregated user with request rate of 25Hz, the request rate for individual mobile user is 5Hz. To simply the simulation results collection, we assume that mobile users only request real-time contents during their movement. This assumption holds as caching performance of on-demand contents is independently related with user mobility since the decision making is based on the popularity of contents. More details of simulation parameter setting can be found in Table 5.1.

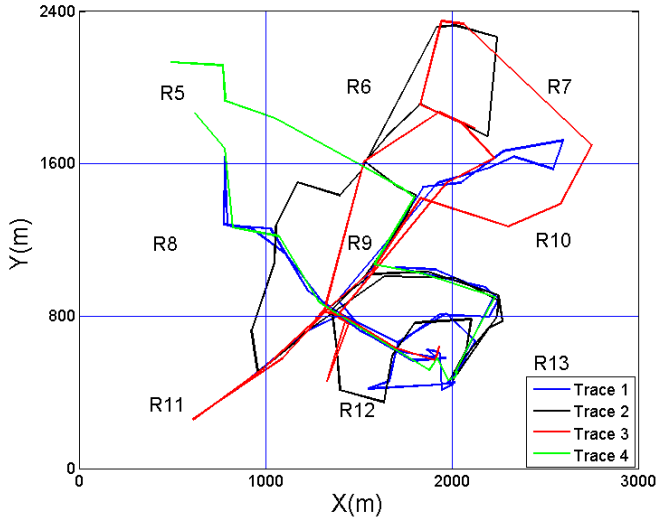


Figure 5.4: Mobility trace in NCSU campus

5.5.2 Performance Metrics

I. One-Hop Cache Hit

One-Hop Cache Hit, as defined in Eq. (5.2), evaluates the hit ratio performance of proactive caching required contents at newly connected routers after handover process. According to the location/identity split feature of CCN architecture, seamless handover in CCN refers to the case that retransmitted requests are satisfied immediately after mobile user switch to newly connected router. Thus, in this metric. only retransmitted requests reach contents in one hop (directly from mobile user) can be considered as a cache hit event. This means that, even the retransmitted requests find the contents in the router located two (or more) hops away, they will not be considered as cache hit. N_h represents the number of cache hit at new connected router and N_r represents the number of retransmitted requests.

$$One\ Hop\ Cache\ Hit = \frac{N_h}{N_r} \tag{5.2}$$

II. Retransmission Hop Distance

Retransmission Hop Distance is the average number of hops that retransmitted requests have to travel to reach the requested contents after handover process. We apply this metric because we are interested in what factors can improve the proactive caching performance as the Retransmission Hop Distance converge to 1. Retransmission Hop Distance equals to 1 is the best case (seamless handover) as requested content can be accessed directly at connected router

after handover. Retransmission Hop Distance is defined in Eq. (5.3), where N_r is the number of retransmitted requests and h_i is the hop distance of retransmitted request.

$$Retransmission\ Hop\ Distance = \frac{\sum_1^{N_r} h_i}{N_r} \tag{5.3}$$

5.5.3 Results of NG-CachePop Decision Policy in Mobile Scenario

In this subsection, we conduct some other simulations in mobile scenario with the purpose for evaluating the fast handover performance of *NG-CachPop*. In our decision policy design, we only proactive caching is only available for requests for real-time content. So in our simulations, we only collect he results from real-time contents, or more specifically, real-time contents during the handover. Intuitively, the caching performance for on-demand contents remains the same if α value of request traffic unchanged. We will first inspect the construction of neighbor graph, which is the fundamental requirement for proactive caching in our design. After that, we analyze the performance of *NG-CachePop* with One-Hop Cache Hit and Retransmission Hop Distance metrics.

Table 5.2: Neighbor Graph Generation Based on Real Traces.

Router ID	Trace 1	Trace 2	Trace 3	Trace 4	Combine
5	8	N/A	N/A	6	6,8
6	N/A	7,9	7,9	5,9	5,7,9
7	10	6	6,9,10	N/A	6,9,10
8	5,9	N/A	N/A	9,	5,9
9	8,10, 12,13	6,10, 12	6,7, 10,12	6,8, 10,12	6,7,8, 10,12,13
10	7,9,13	9,13	7,9	9,13	7,9,13
11	12	9,12	9,12	N/A	9,12
12	9,11,13	9,11 ,13	9,11	9,13	9,11,13
13	9,10,12	10,12	N/A	10,12	9,10,12

Neighbor Graph Generation

Neighbor graph generation is the key for proactive caching in *NG-CachePop* policy. Due to the difference of router locations and mobile traces, the neighbor graph generated at each router might not be exactly the same as physical deployment in real world. Tab. 5.2 provide the neighbor graphes generated at edge routers by applying four different mobile trace individually. Notice that the traces of human movement (as shown in Fig. 5.4) have preference depending

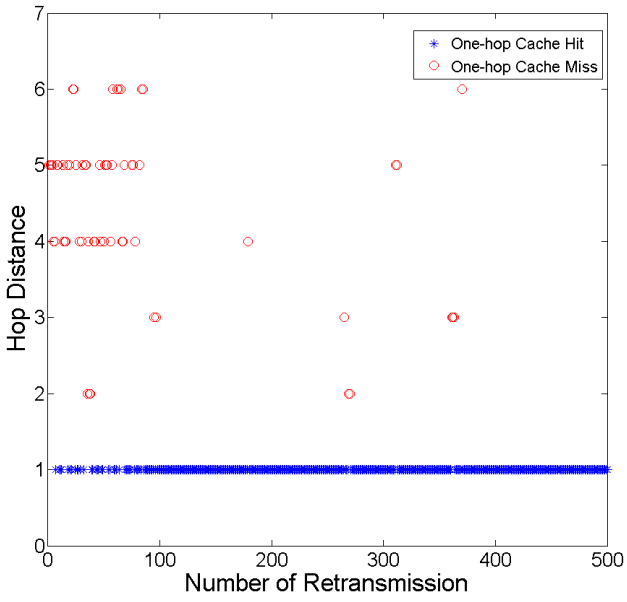


Figure 5.5: Hop distance with number of retransmitted packet

on the road directions or movement preference. Thus, the number of neighbors at each router generated by these traces might be different. For example, router 5 only has one neighbor router 8 and 6 in Trace 1 and 4 respectively, while router 9 has three or more neighbor router at each trace. Although router 6 is located one-hop away from router 5, it is not considered as neighbor for router 5 in trace 1, 2 and 3. Instead of generating a completed pair-wised neighbor graph, *NG-CachPop* generates a more simply and robust neighbor graph to predict the next potential connection of mobile users.

After fully warm up, we obtain final neighbor graph which is the combination of neighbor graph generated from four different traces. Clearly, we can see that the combined neighbor graphs at each routers are more accurate to reflect the real world routers deployment. This result implies that with more traffic traces (mobile users), the neighbor graph construction can be more accurate. It also confirms that neighbor graph approach is feasible for providing potential routers map for proactive caching. In order to reduce the duplicated contents forwarded to neighbor routers, the neighbor graph size should be limited. In Tab. 5.2, neighbor graph size at 3 is enough for most of the routers except router 9. Thus, we set the neighbor graph size to be 3 in the simulation.

The warmup process of neighbor graph generation in Fig. 5.5. We show the hop distance of the first 500 retransmitted requests at edge routers. We define One-hop Cache Hit when retransmitted requests access the content at the connected edge routers after handover. Otherwise, it is

considered as One-hop Cache Miss when the hop distance is larger than 1. At the beginning of simulation, most of the retransmitted requests get cache misses with hop distance up to 4 hop. This is because that neighbor graphs of each routers have not been fully generated yet. After over 80 retransmitted *Interest* packets are send for neighbor graph establishment, the number of cache miss decreases rapidly. As we observe in the graph, most of the retransmitted *Interest* packets get the proactive caching hit except for the very beginning warmup process and some outliers. In fact, the latency introduced by neighbor graph warmup is negligible because it only occurs when routers are added in or removed from the network.

Hit Ratio and Hop Distance Results

We collect the One-Hop Cache Hit and Retransmission Hop Distance results after neighbor graph is fully warmed up. We evaluate the performance of *NG-CachePop* under different mobile user number and mobile cache size. As expected, Fig. 5.6 shows that One-Hop Hit Ratio increases with the increasing mobile cache size since more real-time contents stored in the cache. In addition, large number of mobile users increases the content refreshing rate in mobile cache, which decreases hit ratio performance because proactive caching required contents are replaced by new arrival contents. The One-Hop Hit Ratio in 25 mobile users case is approaching to 1 at mobile cache size larger than 40, indicating that almost all the retransmitted requests are satisfied immediately after handover.

Fig. 5.7 shows the Retransmission Hop Distance results. The minimum hop distance for prefect proactive cache policy is 1, which means the mobile user can access the contents requested before handover directly after connecting to the new PoA. By inspecting the results in Fig. 5.7, we observe that when mobile cache size larger than 20, the average hop distance in 25 mobile users case is very close to 1. Even in 100 mobile user number case, the average hop distance can be decreased to less than 1.5 when mobile cache size larger than 35.

We also compare the *NG-CachePop* with existing caching decision policies (includes *CachePop*) in 50 mobile user mobile scenario. We provide the CDF of hop distance results in Fig.5.8. In hybrid topology, the hop distance from edge router to content server varies from 4 to 6. For policies without proactive caching (e.g., *CacheAll*, *CacheRdm*, *ProbCache*, *CachePop*), probability of hop distance equals or less than 3 is less than 0.2. As a pure popularity-oriented cache policy, *CachePop* suffers the worst case in which handover latency is double RTT since no hop distance is less than 4. In contrast to these CDPs, *NG-CachePop* performs much better for reducing the handover latency of real-time content. Even with very small mobile cache size 10, the probability that retransmitted requests being satisfied at directly connected edge router (hop distance = 1) is about 55%. Over 70% of retransmitted requests can reach contents within the network (hop distance < 4). By increasing the mobile cache size to 20 and 30, the proba-

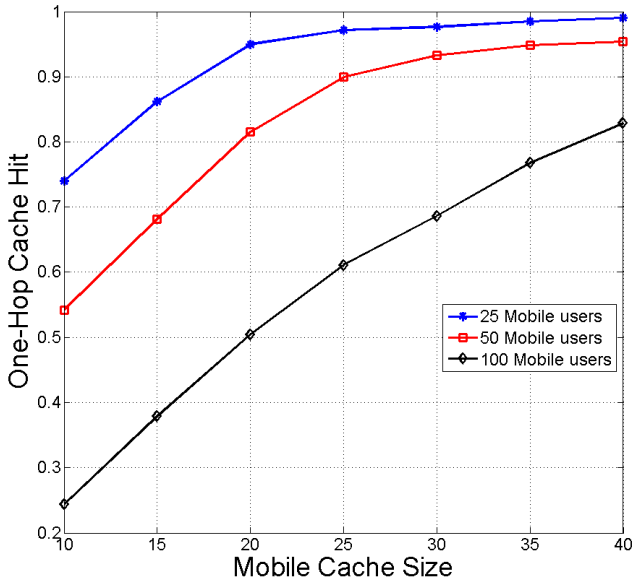


Figure 5.6: One-Hop Cache Hit results with different mobile user and cache size

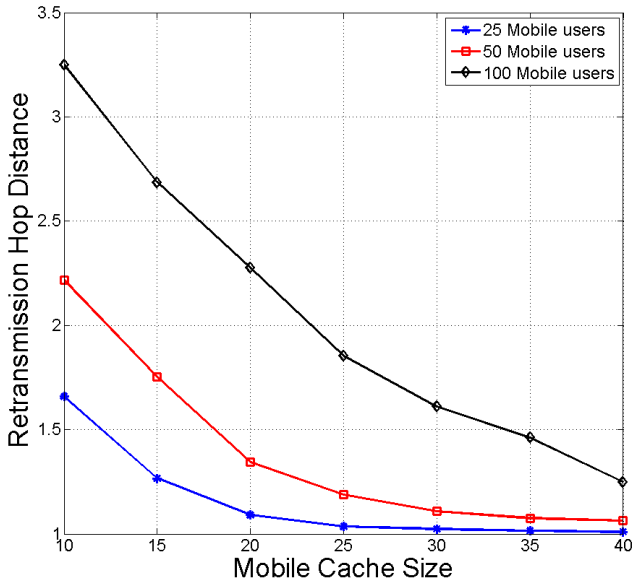


Figure 5.7: Retransmission Hop Distance results with different mobile user and cache size

bility that retransmitted requests access the content in 1 hop distance climbs to over 80% and 90% respectively.

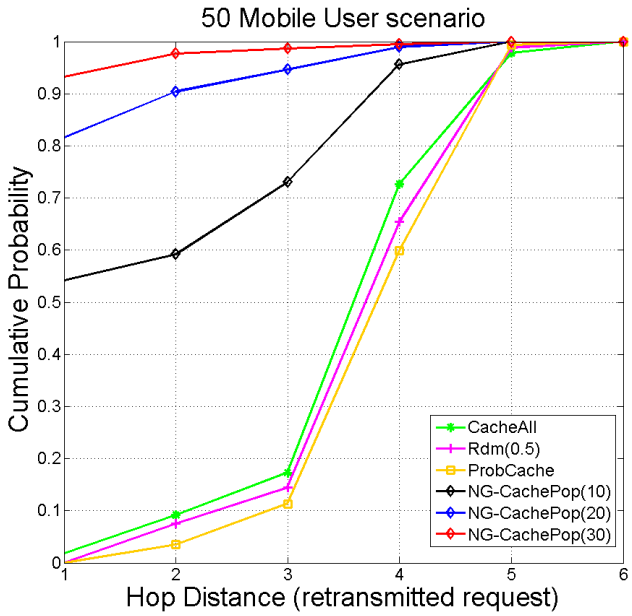


Figure 5.8: Performance comparison for real-time contents in mobile scenario.

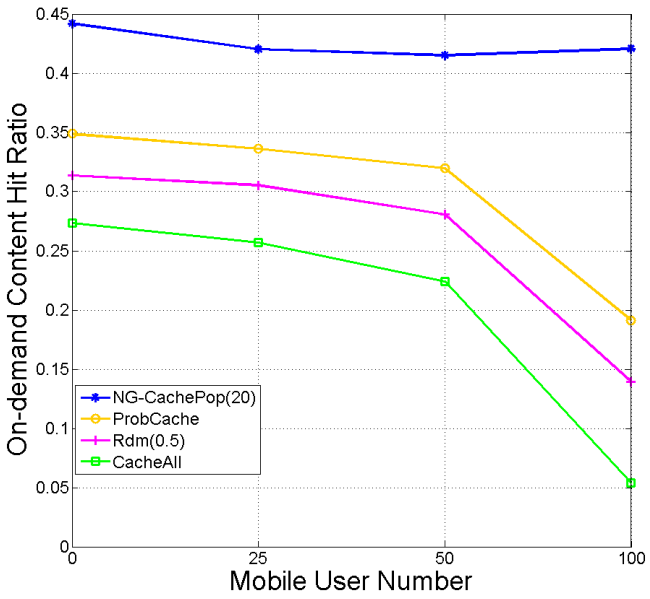


Figure 5.9: Performance comparison for on-demand contents in mobile scenarios.

To avoid content pollution on the router cache, *NG-CachePop* applies two caches for two types content individually. For better result demonstration, we increase the α value from 0.8 to

1.0 in this simulation. The number of mobile user = 0 refers to non-mobile scenario in which the entire cache (CCR=1%) is used for caching on-demand contents. When mobile users are introduced, we set the mobile cache catalog ratio (MCCR)=0.2% and regular cache catalog ratio (RCCR)=0.8% for *NG-CachePop* (the overall CCR still remains to 1%). Simulation results are demonstrated in Fig.5.9. First of all, we can see that *NG-CachePop* outperforms other three cache policies in non-mobile scenario with the hit ratio at 0.45. It is 28% improvement compared with *ProbCache* and 61% improvement compared with *CacheAll* policy. After mobile users are added, the hit ratio performance of all cache policies decrease to some extent. However, the results validate the robustness of *NG-CachePop* as hit ratio of on-demand content remains stable in mobile scenario when number of mobile user increases. On the contrary, as the number of mobile uses increases, the hit ratio performance of all other CDPs decreases accordingly. For instance, when the number of mobile users larger than 100, the hit ratio of *ProbCache* drops below 0.2, which is over 100% smaller than the result (0.43) that *NG-CachePop* brings about. These results show that *NG-CachePop* can guarantee the performance of on-demand contents in both non-mobile and mobile scenarios.

5.6 Summary

In this chapter, we define the neighbor graph generation in CCNs and design the proactive caching scheme in mobile CCN. We propose *NG-CachePop* policy by adding neighbor graph and proactive caching features on *CachePop* policy. In particular, we define two types of contents: on-demand and real-time, and apply different decision policies accordingly in *NG-CachePop*. *NG-CachePop* aims to remain the caching efficiency for on-demand contents and achieve seamless handover for real-time contents. We evaluate our *NG-CachePop* through *MCsim* in various mobile CCN scenarios with real mobility trace obtained in NCSU campus. The neighbor graph generation result confirms that neighbor graph correctly identify the potential next-hop routers based on user mobility. The proactive caching hit ratio result shows that *NG-CachePop* can reach 0.8 in general scenario (50 mobile users, 20% mobile cache size), implying that over 93.3% of retransmission requests can assess the content immediately after handover process.

Chapter 6

Conclusion and Future Works

In this thesis, we have presented our research results regarding the simulation study on CCN caching performance and the cache policy design for improving the caching efficiency and handover latency. Next, we summarize what we have achieved in this research and discuss the future works.

6.1 Conclusion

This thesis research has focused on the in-network caching study and cache policy design in emerging Content Centric Network (CCN). We studied the improvements of CCN on content dissemination and mobility support and identified the inefficiency problem of existing cache policy. Motivated by improving caching efficiency and solve the handover latency problem, we designed a neighbor graph based popularity-oriented cache decision policy *NG-CachePop* in this research. Evaluation results confirm that *NG-CachePop* achieves seamless handover for real-time content in general mobile CCN scenario and improves the hit ratio results for on-demand contents as well. Next, we recapitulate and highlight our major contributions.

In Chapter 2, we provided a detailed review on the caching relevant researches with respect to caching architectures, cache decision and replacement policy among different networks. In addition, we analyzed the related researches on the mobility management in traditional IP network and CCN scenarios. After reviewing the relevant researches on CCN. we find that most of works focus on either the in-network caching performance (especially on cache replacement policy) or mobility support perspective. No work has ever considered how the mobility support designs are compatible with in-network caching schemes. And none of them has ever considered to solve handover latency problem in cache policy aspect. Therefore, our work fills this void in CCN by taking these two issues into consideration.

In Chapter 3, we developed a scalable caching simulator (*MCsim*) which supports multi-

parameters configurability in both static and mobile CCN scenarios for the purpose to provide a comprehensive caching study and evaluate the performance of cache policies in CCN. We thoroughly discussed the network and system settings and define the caching simulation scenarios in our work by making a CCN parameter settings survey on related works in CCN. With our thorough simulation study, we found that the inefficiency of universal caching strategy (*CacheAll*) is due to high replacement error and cache pollution in upstream nodes. We identified that content popularity plays the most critical role in the caching performance. In addition, we showed that cache decision should be made based on arrival requests popularity rather than pure content popularity to avoid cache pollution problem. These observations greatly help for guiding the cache policy design and optimization in CCN.

In Chapter 4, we proposed the popularity-oriented caching decision policy *CachePop* by following the guidelines summarized in our simulation study. In particular, we added a Request Table in CCN architecture for recording the arrival requests information. We specified the optimal Xm valued used in decision making process and Request Table control method. After that, we validated our *CachePop* cache policy through *MCsim* with respect to Cache Hit Ratio, Content Diversity and Hop Distance Ratio. The simulation results showed that *CachePop* significantly improves the caching efficiency in both the node and network level compared to other policies. By inspecting the Content Diversity results, we observed that *CachePop* successfully identifies and caches the most popular contents in arrival requests and achieves hierarchical caching. In addition, simulation result demonstrated that *CachePop* can quickly responds to content popularity change in request traffic.

In Chapter 5, we defined the neighbor graph generation in CCN and design the proactive caching scheme in mobile CCN. We improved the *CachePop* policy by adding these two features to solve mobile latency problem for real-time content and propose *NG-CachePop* to achieve seamless handover and improve content delivery efficiency at the same time. In particular, we defined two types of contents: on-demand and real-time, and applied different decision policies accordingly in *NG-CachePop*. We evaluated our *NG-CachePop* through *MCsim* in various mobile CCN scenarios with real mobility trace obtained in NCSU campus. The neighbor graph generation result confirmed that neighbor graph correctly identify the potential next-hop routers based on user mobility. The One-Hop Cache Hit result showed that *NG-CachePop* can reach 0.8 in general scenario (50 mobile users, 20% mobile cache size), implying that over 93.3% of retransmission requests can assess the content immediately after handover process.

6.2 Future Works

The work presented in this thesis focuses efforts on fast proactive caching to achieve seamless handover. To this end, we forward handover-required contents to a subset of routers which are

considered as next stops that mobile user will connect to. However, this strategy might introduce the security problem such as flooding attack especially when neighbor size is large. Thus, this exposed security issue should be considered in the future research. Possible approaches include neighbor graph size control, handover device registration and limiting the number of handover-required request by users.

In addition, as contents are forwarded to every router in the neighbor graph set, mobile cache in the routers which are not the next stop for mobile user will be occupied inefficiently. Such cache error can degrade the performance of *NG-CachePop*. One possible improvement is to enable invalid cache notification so that router can delete contents in mobile cache if the contents have been delivered to users at another router. Applying the control messages might result in extra bandwidth consumption, but also enhances the mobile cache utilization. Therefore, it is highly desirable to assess the tradeoff between network bandwidth consumption and performance of mobility support in *NG-CachePop*. In order to further evaluate the *NG-CachePop*, we can utilize the real request traffic traces and more mobility traces in the evaluation process.

REFERENCES

[1] Barry M. Leiner, Vinton G. Cerf, David D. Clark, Robert E. Kahn, Leonard Kleinrock, Daniel C. Lynch, Jon Postel, Larry G. Roberts, and Stephen Wolff. A brief history of the internet. *SIGCOMM Comput. Commun. Rev.*, 39(5):22–31, October 2009.

[2] Craig Labovitz, Scott Iekel-Johnson, Danny McPherson, Jon Oberheide, and Farnam Jahanian. Internet inter-domain traffic. *SIGCOMM Comput. Commun. Rev.*, 41(4):–, August 2010.

[3] Bengt Ahlgren, Christian Dannewitz, Claudio Imbrenda, Dirk Kutscher, and Börje Ohlman. A survey of information-centric networking. *Communications Magazine, IEEE*, 50(7):26–36, 2012.

[4] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael Plass, Nick Briggs, and Rebecca Braynard. Networking named content. *Commun. ACM*, 55(1):117–124, January 2012.

[5] Teemu Koponen, Mohit Chawla, Byung-Gon Chun, Andrey Ermolinskiy, Kye Hyun Kim, Scott Shenker, and Ion Stoica. A data-oriented (and beyond) network architecture. In *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '07, pages 181–192, New York, NY, USA, 2007. ACM.

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

[7] Dmitrij Lagutin, Kari Visala, and Sasu Tarkoma. Publish/subscribe for internet: Psirp perspective. *Valencia FIA book*, 4, 2010.

[8] Gerardo García, Andrzej Beben, Francisco J Ramón, Adrián Maeso, Ioannis Psaras, George Pavlou, Ning Wang, Jaroslaw Sliwinski, Spiros Spirou, Sergios Soursos, et al. Comet: content mediator architecture for content-aware networks. In *Future Network & Mobile Summit (FutureNetw), 2011*, pages 1–8. IEEE, 2011.

[9] J. Choi, J. Han, E. Cho, T. Kwon, and Y. Choi. A survey on content-oriented networking for efficient content delivery. *Communications Magazine, IEEE*, 49(3):121–127, 2011.

[10] Yusung Kim and Ikjun Yeom. Performance analysis of in-network caching for content-centric networking.

[11] Wei Koong Chai, Diliang He, Ioannis Psaras, and George Pavlou. Cache "less for more" in information-centric networks. In *Proceedings of the 11th international IFIP TC 6 conference on Networking - Volume Part I*, IFIP'12, pages 27–40, Berlin, Heidelberg, 2012. Springer-Verlag.

[12] Ioannis Psaras, Wei Koong Chai, and George 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, pages 55–60, New York, NY, USA, 2012. ACM.

[13] Zhe Li, Gwendal Simon, and Annie Gravey. Caching policies for in-network caching. In *Computer Communications and Networks (ICCCN), 2012 21st International Conference on*, pages 1–7. IEEE, 2012.

[14] Sen Wang, Jun Bi, Jianping Wu, Zhaogeng Li, Wei Zhang, and Xu Yang. Could in-network caching benefit information-centric networking? In *Proceedings of the 7th Asian Internet Engineering Conference*, AINTEC '11, pages 112–115, New York, NY, USA, 2011. ACM.

[15] Dario Rossi and Giuseppe Rossini. Caching performance of content centric networks under multi-path routing (and more). 2011.

[16] D.R.G. Rossini and D. Rossi. A dive into the caching performance of content centric networking. Technical report, Technical report, Telecom ParisTech, 2011.

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

[18] Ravishankar Ravindran, Samantha Lo, Xinwen Zhang, and Guoqiang Wang. Supporting seamless mobility in named data networking. In *Communications (ICC), 2012 IEEE International Conference on*, pages 5854–5869. IEEE, 2012.

[19] Xenofon Vasilakos, Vasilios A. Siris, George C. Polyzos, and Marios 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, pages 61–66, New York, NY, USA, 2012. ACM.

[20] Gareth Tyson, Nishanth Sastry, Ivica Rimac, Ruben Cuevas, and Andreas Mauthe. A survey of mobility in information-centric networks: challenges and research directions. In *Proceedings of the 1st ACM workshop on Emerging Name-Oriented Mobile Networking Design - Architecture, Algorithms, and Applications*, NoM '12, pages 1–6, New York, NY, USA, 2012. ACM.

[21] D. Kim, J. Kim, Y. Kim, H. Yoon, and I. Yeom. Mobility support in content centric networks. In *Proceedings of the second edition of the ICN workshop on Information-centric networking*, pages 13–18. ACM, 2012.

[22] V. Sourlas, G.S. Paschos, P. Flegkas, and L. Tassiulas. Mobility support through caching in content-based publish/subscribe networks. In *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, pages 715–720. IEEE, 2010.

[23] J. Wang, J. Cao, J. Li, and J. Wu. Mhh: A novel protocol for mobility management in publish/subscribe systems. In *Parallel Processing, 2007. ICPP 2007. International Conference on*, pages 54–54. IEEE, 2007.

[24] Rim Haw and Choong Seon Hong. A seamless content delivery scheme for flow mobility in content centric network. In *Network Operations and Management Symposium (APNOMS), 2012 14th Asia-Pacific*, pages 1–5. IEEE, 2012.

[25] Jihoon Lee, Sungrae Cho, and Daeyoub Kim. Device mobility management in content-centric networking. *Communications Magazine, IEEE*, 50(12):28–34, 2012.

[26] Yunqi Luo, Jonas Eymann, Kishore Angrishi, and Andreas Timm-Giel. Mobility support for content centric networking: Case study. In *Mobile Networks and Management*, pages 76–89. Springer, 2012.

[27] Tobias Lauinger, Nikolaos Laoutaris, Pablo Rodriguez, Thorsten Strufe, Ernst Biersack, and Engin Kirda. Privacy risks in named data networking: what is the cost of performance? *ACM SIGCOMM Computer Communication Review*, 42(5):54–57, 2012.

[28] Mauro Conti, Paolo Gasti, and Marco Teoli. A lightweight mechanism for detection of cache pollution attacks in named data networking. *Computer Networks*, 2013.

[29] Mengjun Xie, Indra Widjaja, and Haining Wang. Enhancing cache robustness for content-centric networking. In *INFOCOM*, pages 2426–2434, 2012.

[30] Van Jacobson, Diana K Smetters, Nicholas H Briggs, Michael F Plass, Paul Stewart, James D Thornton, and Rebecca L Braynard. Voccn: voice-over content-centric networks. In *Proceedings of the 2009 workshop on Re-architecting the internet*, pages 1–6. ACM, 2009.

[31] Somaya Arianfar, Pekka Nikander, and Jörg Ott. On content-centric router design and implications. In *Proceedings of the Re-Architecting the Internet Workshop*, page 5. ACM, 2010.

[32] Christine Fricker, Philippe Robert, James Roberts, and Nada Sbihi. Impact of traffic mix on caching performance in a content-centric network. In *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, pages 310–315. IEEE, 2012.

[33] Jia Wang. A survey of web caching schemes for the internet. *ACM SIGCOMM Computer Communication Review*, 29(5):36–46, 1999.

[34] Terence Kelly, Yee Man Chan, Sugih Jamin, and Jeffrey MacKie-Mason. Biased replacement policies for web caches: Differential quality-of-service and aggregate user value. In *Proceedings of the 4th International Web Caching Workshop*, 1999.

[35] K. Psounis and B. Prabhakar. A randomized web-cache replacement scheme. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1407–1415. IEEE, 2001.

[36] Wei Gao, Guohong Cao, Arun Iyengar, and Mudhakar Srivatsa. Supporting cooperative caching in disruption tolerant networks. In *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, pages 151–161. IEEE, 2011.

[37] Tiance Wang, Pan Hui, Sanjeev R Kulkarni, and Paul Cuff. Cooperative caching based on file popularity ranking in delay tolerant networks. 2012.

[38] Xuejun Zhuo, Qinghua Li, Guohong Cao, Yiqi Dai, Boleslaw Szymanski, and Tom La Porta. Social-based cooperative caching in dtns: a contact duration aware approach. In *Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on*, pages 92–101. IEEE, 2011.

[39] Jian Ni and Danny HK Tsang. Large-scale cooperative caching and application-level multicast in multimedia content delivery networks. *Communications Magazine, IEEE*, 43(5):98–105, 2005.

[40] George Pallis and Athena Vakali. Insight and perspectives for content delivery networks. *Communications of the ACM*, 49(1):101–106, 2006.

[41] S. Borst, V. Gupta, and A. Walid. Distributed caching algorithms for content distribution networks. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE, 2010.

[42] Yan Chen, Randy H Katz, and John D Kubiatowicz. Dynamic replica placement for scalable content delivery. In *Peer-to-Peer Systems*, pages 306–318. Springer, 2002.

[43] L. Yin and G. Cao. Supporting cooperative caching in ad hoc networks. *Mobile Computing, IEEE Transactions on*, 5(1):77–89, 2006.

[44] N. Chand, R.C. Joshi, and M. Misra. Efficient cooperative caching in ad hoc networks. In *Communication System Software and Middleware, 2006. Comsware 2006. First International Conference on*, pages 1–8. IEEE, 2006.

[45] N. Chand, RC Joshi, and M. Misra. Cooperative caching strategy in mobile ad hoc networks based on clusters. *Wireless Personal Communications*, 43(1):41–63, 2007.

[46] Duc A Tran, Minh Le, and Kien A Hua. Mobivod: a video-on-demand system design for mobile ad hoc networks. In *Mobile Data Management, 2004. Proceedings. 2004 IEEE International Conference on*, pages 212–223. IEEE, 2004.

[47] H. Chen and Y. Xiao. Cache access and replacement for future wireless internet. *Communications Magazine, IEEE*, 44(5):113–123, 2006.

[48] S. Podlipnig and L. Böszörmenyi. A survey of web cache replacement strategies. *ACM Computing Surveys (CSUR)*, 35(4):374–398, 2003.

[49] Hao Che, Zhijung Wang, and Ye Tung. Analysis and design of hierarchical web caching systems. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1416–1424. IEEE, 2001.

[50] Y Thomas Hou, Jianping Pan, Bo Li, Xueyan Tang, and Shivendra Panwar. Modeling and analysis of an expiration-based hierarchical caching system. In *Global Telecommunications Conference, 2002. GLOBECOM'02. IEEE*, volume 3, pages 2468–2472. IEEE, 2002.

[51] Ali Ghodsi, Scott Shenker, Teemu Koponen, Ankit Singla, Barath Raghavan, and James Wilcox. Information-centric networking: seeing the forest for the trees. In *Proceedings of*

the 10th ACM Workshop on Hot Topics in Networks, HotNets-X, pages 1:1–1:6, New York, NY, USA, 2011. ACM.

[52] Nikolaos Laoutaris, Hao Che, and Ioannis Stavrakakis. The lcd interconnection of lru caches and its analysis. *Performance Evaluation*, 63(7):609–634, 2006.

[53] Konstantinos Katsaros, George Xylomenos, and George C Polyzos. Multicache: An overlay architecture for information-centric networking. *Computer Networks*, 55(4):936–947, 2011.

[54] Lada A Adamic and Bernardo A Huberman. Zipfs law and the internet. *Glottometrics*, 3(1):143–150, 2002.

[55] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web caching and zipf-like distributions: Evidence and implications. In *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 126–134. IEEE, 1999.

[56] Mobile vs. desktop traffic from Dec. 2008 to Jan. 2013. `http://gs.statcounter.com/#mobile_vs_desktop-ww-monthly-200901-201212`.

[57] Global Mobile Data Traffic Forecast Update 2012 to 2017. *Cisco Visual Networking Index*.

[58] C. Spanner P. Rodriguez and E. Biersack. Analysis of web caching architectures: Hierarchical and distributed caching. *Networking, IEEE/ACM Transactions on*, 9(4):401–418, 2001.

[59] Al-Mukaddim Khan Pathan and Rajkumar Buyya. A taxonomy and survey of content delivery networks. *Grid Computing and Distributed Systems Laboratory, University of Melbourne, Technical Report*, 2007.

[60] Anawat Chankhunthod, Peter B Danzig, Chuck Neerdaels, Michael F Schwartz, and Kurt J Worrell. A hierarchical internet object cache. Technical report, DTIC Document, 1995.

[61] Scott Michel, Khoi Nguyen, Adam Rosenstein, Lixia Zhang, Sally Floyd, and Van Jacobson. Adaptive web caching: towards a new global caching architecture. *Computer Networks and ISDN systems*, 30(22):2169–2177, 1998.

[62] Gang Peng. Cdn: Content distribution network. *CoRR*, cs.NI/0411069, 2004.

[63] Athena Vakali and George Pallis. Content delivery networks: Status and trends. *Internet Computing, IEEE*, 7(6):68–74, 2003.

[64] Duane Wessels. Application of internet cache protocol (icp), version 2. 1997.

[65] Alex Rousskov and Duane Wessels. Cache digests. *Computer Networks and ISDN Systems*, 30(22):2155–2168, 1998.

[66] Li Fan, Pei Cao, Jussara Almeida, and Andrei Z Broder. Summary cache: a scalable wide-area web cache sharing protocol. *IEEE/ACM Transactions on Networking (TON)*, 8(3):281–293, 2000.

[67] Syam Gadde, Michael Rabinovich, and Jeff Chase. Reduce, reuse, recycle: An approach to building large internet caches. In *Operating Systems, 1997., The Sixth Workshop on Hot Topics in*, pages 93–98. IEEE, 1997.

[68] Marc Abrams, Charles R Standridge, Ghaleb Abdulla, Stephen Williams, and Edward A Fox. Caching proxies: Limitations and potentials. 1995.

[69] Marc Abrams, Charles R Standridge, Ghaleb Abdulla, Edward A Fox, and Stephen Williams. Removal policies in network caches for world-wide web documents. *ACM SIGCOMM Computer Communication Review*, 26(4):293–305, 1996.

[70] Giovanna Carofiglio, Vinicius Gehlen, and Diego Perino. Experimental evaluation of memory management in content-centric networking. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–6. IEEE, 2011.

[71] Martin Arlitt, Ludmila Cherkasova, John Dilley, Rich Friedrich, and Tai Jin. Evaluating content management techniques for web proxy caches. *ACM SIGMETRICS Performance Evaluation Review*, 27(4):3–11, 2000.

[72] Terence Kelly, Yee Man Chan, Sugih Jamin, and Jeffrey MacKie-Mason. Biased replacement policies for web caches: Differential quality-of-service and aggregate user value. In *Proceedings of the 4th International Web Caching Workshop*, 1999.

[73] Bo Wang, Yong Wei Wu, and Wei Min Zheng. Web caching replacement based on user's visiting action. *Applied Mechanics and Materials*, 52:25–30, 2011.

[74] Chi-Feng Kao and Chung-Nan Lee. Aggregate profit-based caching replacement algorithms for streaming media transcoding proxy systems. *Multimedia, IEEE Transactions on*, 9(2):221–230, 2007.

[75] Zhongxing Ming, Mingwei Xu, and Dan Wang. Age-based cooperative caching in information-centric networks. In *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, pages 268–273. IEEE, 2012.

[76] Bin Tang, Himanshu Gupta, and Samir R Das. Benefit-based data caching in ad hoc networks. *Mobile Computing, IEEE Transactions on*, 7(3):289–304, 2008.

[77] S Ayyasamy and SN Sivanandam. A qos-aware intelligent replica management architecture for content distribution in peer-to-peer overlay networks. *arXiv preprint arXiv:0912.2296*, 2009.

[78] Sharrukh Zaman and Daniel Grosu. A distributed algorithm for web content replication. In *Network Computing and Applications, 2009. NCA 2009. Eighth IEEE International Symposium on*, pages 284–287. IEEE, 2009.

[79] Nikolaos Laoutaris, Orestis Telelis, Vassilios Zissimopoulos, and Ioannis Stavrakakis. Distributed selfish replication. *Parallel and Distributed Systems, IEEE Transactions on*, 17(12):1401–1413, 2006.

[80] Nikolaos Laoutaris, Sofia Syntila, and Ioannis Stavrakakis. Meta algorithms for hierarchical web caches. In *Performance, Computing, and Communications, 2004 IEEE International Conference on*, pages 445–452. IEEE, 2004.

[81] Charles E Perkins. Mobile ip. *Communications Magazine, IEEE*, 35(5):84–99, 1997.

[82] Eva Fogelstroem, Annika Jonsson, and Charles E Perkins. Mobile ipv4 regional registration. 2007.

[83] Jonathan Rosenberg, Henning Schulzrinne, Gonzalo Camarillo, Alan Johnston, Jon Peterson, Robert Sparks, Mark Handley, Eve Schooler, et al. Sip: session initiation protocol. Technical report, RFC 3261, Internet Engineering Task Force, 2002.

[84] Robert Hsieh, Zhe Guang Zhou, and Aruna Seneviratne. S-mip: A seamless handoff architecture for mobile ip. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 1774–1784. IEEE, 2003.

[85] Luca Muscariello, Giovanna Carofiglio, and Massimo Gallo. Bandwidth and storage sharing performance in information centric networking. In *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, pages 26–31. ACM, 2011.

[86] Giovanna Carofiglio, Massimo Gallo, Luca Muscariello, and Diego Perino. Modeling data transfer in content-centric networking. In *Proceedings of the 23rd International Teletraffic Congress*, pages 111–118. ITCP, 2011.

[87] Ioannis Psaras, Richard G Clegg, Raul Landa, Wei Koong Chai, and George Pavlou. Modelling and evaluation of ccn-caching trees. In *NETWORKING 2011*, pages 78–91. Springer, 2011.

[88] Meeyoung Cha, Haewoon Kwak, Pablo Rodriguez, Yong-Yeol Ahn, and Sue Moon. I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 1–14. ACM, 2007.

[89] Arunesh Mishra, Minho Shin, and WA Arbaush. Context caching using neighbor graphs for fast handoffs in a wireless network. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 1. IEEE, 2004.

[90] Chung-Sheng Li, Yung-Chih Tseng, and Han-Chieh Chao. A neighbor caching mechanism for handoff in ieee 802.11 wireless networks. In *Multimedia and Ubiquitous Engineering, 2007. MUE'07. International Conference on*, pages 48–53. IEEE, 2007.

[91] Sang-Hee Park, Hye-Soo Kim, Chun-Su Park, Jae-Won Kim, and Sung-Jea Ko. Selective channel scanning for fast handoff in wireless lan using neighbor graph. In *Personal Wireless Communications*, pages 194–203. Springer, 2004.

[92] Injong Rhee, Minsu Shin, Seongik Hong, Kyunghan Lee, Seongjoon Kim, and Song Chong. CRAWDAD data set ncsu/mobilitymodels (v. 2009-07-23). Downloaded from http://crawdad.cs.dartmouth.edu/ncsu/mobilitymodels, July 2009.