Doctoral Dissertation

A Study on Clustering Schemes for Efficient Information Distribution in Information-Centric Networking

Mikiya Yoshida

March, 2024

Graduate School of Environmental Engineering
The University of Kitakyushu

Preface

The primary usage of today's network is for delivering video content, such as Video on Demand (VoD), which constitutes approximately 80% of the total network traffic. It is anticipated that in the near future, the proliferation of IoT (Internet of Things) services will lead to the influx of a large volume of IoT content into the network. Therefore, solutions are needed to support the current network usage, which revolves around content distribution. A new network architecture called Information-Centric Networking (ICN) has attracted attention as a potential solution. ICN is designed with a focus on the concept that, when consumers access content, they are indifferent to the identity of the content provider. This design allows for directly exploring content in the network without being dependent on the content provider by changing the destination of a content request from the traditional Internet protocol (IP) address to the content name. To take advantage of this design concept, content routers (CRs), which are intermediate routers, are assigned the roles of both forwarding and caching content. This allows replicated content to be distributed into the network while acting as substitutes for content providers. Therefore, ICN can handle the content requests by using CRs without the intervention of content providers, which contributes to balancing server and network loads and reducing content delivery latency.

However, the cache size of CR is significantly smaller than the huge amount of content worldwide, so it is naturally impossible to cache all content. Therefore, an important issue to be addressed by ICN is to consider how to use these cashes, since they significantly affect the performance of content retrieval. In recent years, to address this issue, clustering schemes have been proposed. These schemes group several CRs (i.e., a cluster) in the network and apply efficient content placement and routing within each cluster. This allows for resolving the majority of consumer content requests by utilizing a near cluster. However, I believe that

we can achieve higher efficiency by focusing on two points, which are still open for discussion: 1. How to efficiently use surrounding clusters, and 2. How to adapt to changes in content demand trends.

Therefore, in this dissertation, as one solution for efficiently using caches in ICN to meet the recent demand for content distribution, I propose a clustering scheme that considers issues 1 and 2. This proposed scheme efficiently delivers content by making clusters that are efficiently clustered and distributed content in accordance with the network situation and by flexibly exploring valuable content from the clusters. Furthermore, the effectiveness of the proposed scheme is demonstrated through simulation evaluations.

The dissertation consists of six chapters. Chapter 1introduces the background, objectives, and structure of the dissertation.

Chapter 2 provides an overview of the inception of ICN, its fundamental operations, related works, and the two specific issues addressed in this dissertation.

Chapter 3 focuses on issue 1 and proposes an efficient content distribution and exploration scheme for clustering schemes. This scheme uniformly distributes content to each cluster in a distributed caching manner and dynamically updates the routing table on the basis of collaboration among CRs across the cluster boundary. As a result, consumers can explore the nearest content from surrounding clusters. In simulation evaluations, the proposed scheme demonstrated higher cache efficiency and lower content delivery latency than conventional schemes by using surrounding clusters/caches.

Chapter 4 focuses on issue 2 and proposes a dynamic clustering scheme to adjust the cluster size in accordance with fluctuations in content demand trends. This scheme effectively estimates the appropriate cluster size by using a simple threshold-based algorithm on the basis of the frequency of cache updates in the cluster. As a result, it enables the construction of consistently appropriate clusters to adjust to shifting content demand trends. Simulation evaluations indicate that the proposed scheme reduces delivery latency while consistently maintaining a high cache efficiency in an environment with changing content demand trends.

Chapter 5 delves into the adaptability of the proposed scheme in practical networks. While the design and effectiveness of the proposed scheme are discussed on the basis of the simple topology in Chapters 3 and 4, applying the scheme to

practical networks requires establishing a clustering scheme for their networks and evaluating its performance regarding topology dependence. Therefore, I propose a clustering scheme for practical networks, and in simulation evaluations, the proposed scheme indicated sufficient applicability in practical networks.

Chapter 6 concludes the study and addresses future work. By deploying the proposed scheme on the network, clusters automatically formed on the basis of content demand trends among consumers will contribute to improving quality of service regarding diverse content distribution in the future.

Acknowledgements

I would like to acknowledge the support and encouragement received from a number of peoples for several years.

First of all, I wish to express my sincere appreciation to Associate Professor Hiroyuki Koga of the University of Kitakyushu. His constant encouragement, guidance through this research, invaluable discussions and advice have greatly helped in accomplishing the research. I also thank him for his careful reading of all papers on the research.

I would also like to express my gratitude to Associate Professor Yurino Sato of the National Institute of Technology, Sasebo College, and Lecturer Yusuke Ito of the University of Kitakyushu for their valuable comments, time and help in completing the research. Their steady support has greatly helped my study.

I wish to thank Professor Takeshi Ikenaga and Associate Professor Daiki Nobayashi of the Kyushu Institute of Technology for their valuable comments, time and help in completing the research.

I am very grateful to Professor Satoshi Uehara, Professor Yasushi Yamazaki, and Associate Professor Hiroki Cho of the University of Kitakyushu for their advice and comments.

I extend thanks to all other members of the Network Engineering Research Group at the University of Kitakyushu for their kindly supports and valuable discussions.

Finally, my greatest appreciation goes to my family. They perpetually helped me whenever I faced various problems. Their long standing has enabled me to complete my degree.

Contents

List of Figures

List of Tables

1 Introduction

It has been almost 60 years since the birth of ARPANET, the prototype of the Internet [1]. It was initially designed for host-to-host communication, which was primarily used for email. In 1990, the World Wide Web system was released, making the Internet widely availability to the general public [2]. Since then, network technology has advanced significantly, and network usage patterns have shifted dramatically. The primary usage of today's network is for delivering video content, such as Video on Demand (VoD), which constitutes approximately 80% of the total network traffic [3]. In the near future, Internet of Things (IoT) services, including big data analysis and artificial intelligence (AI), are expected to proliferate significantly to realize self-driving cars, industrial robots, etc [4].

With these shifts in network usage patterns, the current network design of host-to-host communication using Internet protocol (IP) addresses faces challenges to satisfying the quality demands for modern services. For example, VoD services, which consist of communication between many consumers and a single producer, cause issues with not only the processing load on the producer side for managing the large number of connections but also the network load from the concentrated flows generated by these connections, as shown in Fig. 1.1. In IoT services such as automated driving, a server collects and processes a large amount of information and provides processed results as control information to vehicles. This service has the problem of communication between the server and a large number of IoT devices that occurs when collecting large amounts of data, i.e., similar to the VoD service, as well as the problem of completing the whole process in an extremely short time (a few milliseconds). Therefore, while communication latency needs to be reduced, the latency caused by the physical distance between the server and the device makes the problem even more difficult. In recent years, the former problem has been solved by Content Delivery Network (CDN) technology [5] and

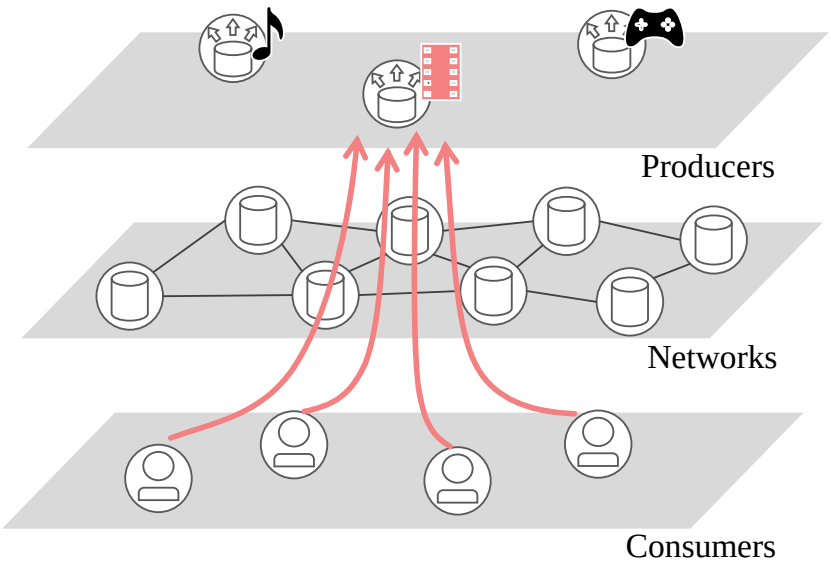


Figure 1.1: IP communication model

the latter by Edge Computing technology [7]. Both solutions commonly deploy additional servers nearby consumers. However, considering the increasing content distribution as various services are introduced in the future, the approach of placing servers for problem resolution on the basis of IP-based architecture each time may not be wise. Therefore, solutions are needed to support the current network usage, which revolves around content distribution.

1.1 Information-Centric Networking (ICN)

A new network architecture called Information-Centric Networking (ICN) has attracted attention as a potential solution [8, 9]. ICN is designed with a focus on the concept that, when consumers access content, they are indifferent to the identity of the content provider. This design allows for directly exploring content in the network without being dependent on the content provider by changing the destination of a content request from the traditional IP address to the content name. To take advantage of this design concept, content routers (CRs), which are intermediate routers, are assigned the roles of both forwarding and caching content. This allows for reducing communication delays caused by physical dis-

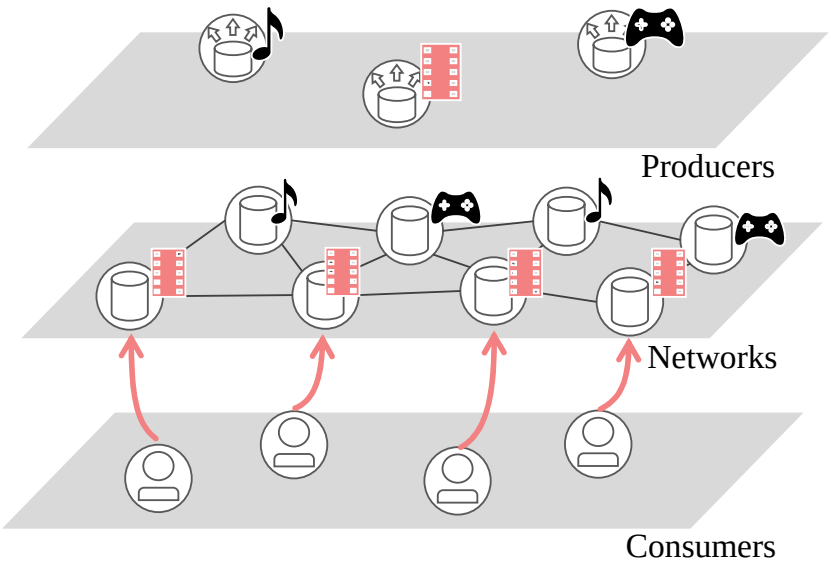


Figure 1.2: ICN communication model

tance, as the requested content from the consumer is responded to using caches at nearby CRs along the route to the producer, as shown in Fig. 1.2. Additionally, CRs distribute the server and network loads by replicating the content on the network (i.e., essentially increasing the number of producers). Specifically, ICN represents a significant departure from conventional approaches by entrusting the network layer (L2) with the resolution of challenges related to content distribution, without the need for additional functionalities such as edge servers or CDN servers. This marks a distinctive paradigm shift, and it is considered a technology that should be realized.

1.2 Issues with ICN

However, the cache size of CR is significantly smaller than the huge amount of content worldwide, so it is naturally impossible to cache all content. Therefore, an important issue to be addressed by ICN is to consider how to use these cashes, since they significantly affect the performance of content retrieval, and many efficient content-caching schemes have been proposed [10–12].

The first step in these schemes is retaining the popular content, i.e., content

frequently requested by consumers, in the cache of a single CR. However, the single-CR approach cannot take into account the cache status of neighboring CRs, which causes cache redundancy. This problem leads to a heavy duplication of popular content in the network and decreases the efficiency of cache utilization. To solve this problem, several schemes have been proposed that organize the caches on the shortest path between the consumer and producer by using probability when determining what to cache. While the above schemes can achieve significant improvements, the cache size (even including caches on the shortest path) is still tiny compared to the huge amount of content worldwide. As a result, off-path caching schemes have been proposed to search caches outside the shortest path by extending the cache exploration range with an additional routing table. However, simply combining it with the caching schemes described above does not achieve sufficient improvement because of redundant caches within the extended cache exploration range.

Therefore, routing and caching should be considered in an integrated manner, and the cache should be placed efficiently within the cache exploration range. In recent years, the exploration range has been replaced by the term "cluster", and clustering schemes [13–16] have been proposed. These schemes group several CRs (i.e., a cluster) in the network and apply efficient content placement and routing within each cluster. This allows for resolving the majority of consumer content requests by utilizing a near cluster. However, I believe that we can achieve higher efficiency by focusing on two points, which are still open for discussion:

1. How to efficiently use surrounding clusters. The routing design of these clustering schemes has not allowed forwarding outside the cluster for request resolution. Specifically, the cache exploration range is restricted to the cluster boundaries, and the requests for non-cached content within the cluster are forwarded directly to the producer. However, although a cluster has a large cache size, it cannot cache the huge amount content worldwide, so the caches in neighboring clusters should also be made available. Therefore, routing that is not restricted by cluster boundaries is necessary.

2. How to adapt to changes in content demand trends. The design of these clustering schemes aims to improve cache utilization and reduce delivery latency by retaining a sufficient amount of the main popular content in the caches of a

cluster. However, in a practical environment, the content demand trends, i.e., the amount of the main popular content, will change over time [17]. Therefore, the cluster size needs to be determined depending on the situation.

1.3 Overview of this dissertation

In this dissertation, as one solution for efficiently using caches in ICN to meet the recent demand for content distribution, I propose a clustering scheme that considers issues 1 and 2. This proposed scheme efficiently delivers contents by making clusters that are efficiently clustered and distributed content in accordance with the network situation and by flexibly exploring valuable content from the clusters. Furthermore, I demonstrate the effectiveness of the proposed scheme through simulation evaluations.

In Chapter 2, I provide an overview of the inception of ICN, its fundamental operations, related works, and the two specific issues addressed in this dissertation.

In Chapter 3, I discuss how to solve issue 1. I propose an efficient distributed caching and exploration scheme for clustering schemes. This scheme uniformly distributes content to each cluster in a distributed caching manner and dynamically updates the routing table on the basis of collaboration among CRs across the cluster boundary. Furthermore, I discuss the effectiveness of the proposed scheme through a simulation evaluation.

In Chapter 4, I discuss how to solve issue 2. I propose a dynamic clustering scheme to adjust the cluster size in accordance with fluctuations in content demand trends. This scheme effectively estimates the appropriate cluster size by using a simple threshold-based algorithm on the basis of the frequency of cache updates in the cluster. Furthermore, I discuss the effectiveness of the proposed scheme through a simulation evaluation that assumes a network with fluctuating content demand trends.

In Chapter 5, I delve into the adaptability of the proposed scheme in practical networks. While the design and effectiveness of the proposed scheme are discussed on the basis of the simple topology in Chapters 3 and 4, applying the scheme to practical networks requires establishing a clustering scheme for their networks and

evaluating its performance regarding topology dependence. Therefore, I propose a clustering scheme for practical networks and discuss its applicability in practical networks through simulation evaluations.

In Chapter 6, I conclude this study and address future work.

The results discussed in Chapter 3 are mainly taken from [18] and those in Chapter 4 and 5 from [19–21].

2 Related works

In this chapter, I provide an overview of the inception of ICN, its fundamental operations, related works, and the two specific issues 1 and 2 addressed in this dissertation.

2.1 Inception of ICN

Today's network usage patterns have changed dramatically from the host-to-host communication of email-like information exchange to the distribution and retrieval of large volumes of content such as Video on Demand (VoD). Since the current network is designed for host-to-host communication, this gap between usage patterns has caused various negative effects [4, 10–12]. When delivering content on the host-to-host communication model on the basis of the IP address, known as IP-network architecture, numerous consumers centrally connect to a single producer, which often causes server load and network congestion. In addition, while delay-sensitive services such as automated driving are expected to be realized, they face a problem related to latency caused by physical distance between producer and consumer. These indicate that the communication model in which performance related to content delivery is highly dependent on the location of the producer and consumer is not appropriate for current network usage patterns.

There have been several technologies to solve this problem on the host-to-host communication model, such as Contents Delivery Network (CDN) [5], Peer-to-Peer (P2P) [5,6], and edge computing [7]. CDNs serve content such as web pages and videos to consumers from servers that are physically close to the consumers, aiming to distribute loads on the servers and enable large data transfers. The fundamental concept of a CDN is that content providers replicate their content

on distributed servers. Based on the physical location of consumers, the request is forwarded through the CDN provider's DNS to the nearest server that holds a replica of the requested content. P2P efficiently distributes content without content providers through a mechanism that allows consumers to easily share downloaded content with others. Namely, it is on the basis of content exchange between consumers, where each consumer can be a consumer as well as a producer. The content requests in P2P networks are resolved by multiple transfers from numerous consumers who already have the content. Edge computing is a solution using edge servers located at the edge of a network to satisfy the requirements of delay-sensitive services. In edge computing, edge servers physically located in the neighborhood process delay-sensitive requests that cloud processing cannot handle. The common idea of these technologies is to place additional servers that can perform the role of producers in the neighborhood of the consumers, i.e., the solutions are on the basis of the application layer. However, considering the increasing content distribution with the deployment of various services in the future, the approach of placing servers for problem resolution each time may not be wise. Therefore, there is a need for solutions to support the current network usage, which revolves around content distribution.

Therefore, Information-Centric Networking (ICN) has been proposed as a new network architecture that solves the location dependent problem in accordance with the current network usage patterns [8]. ICN is designed with a focus on the concept that, when consumers access content, they are indifferent to the identity of the content provider. This design allows for directly exploring content in the network without being dependent on the content provider by changing the destination of a content request from the traditional IP address to the content name. To take advantage of this design concept, content routers (CRs), which are intermediate routers, are assigned the roles of both forwarding and caching content. This allows for reducing communication delays caused by physical distance, as the requested content from the consumer is responded to using caches at nearby CRs along the route to the producer. Additionally, CRs distribute the server and network loads by replicating the content on the network (i.e., essentially increasing the number of producers). Specifically, ICN represents a significant departure from conventional approaches by entrusting the network

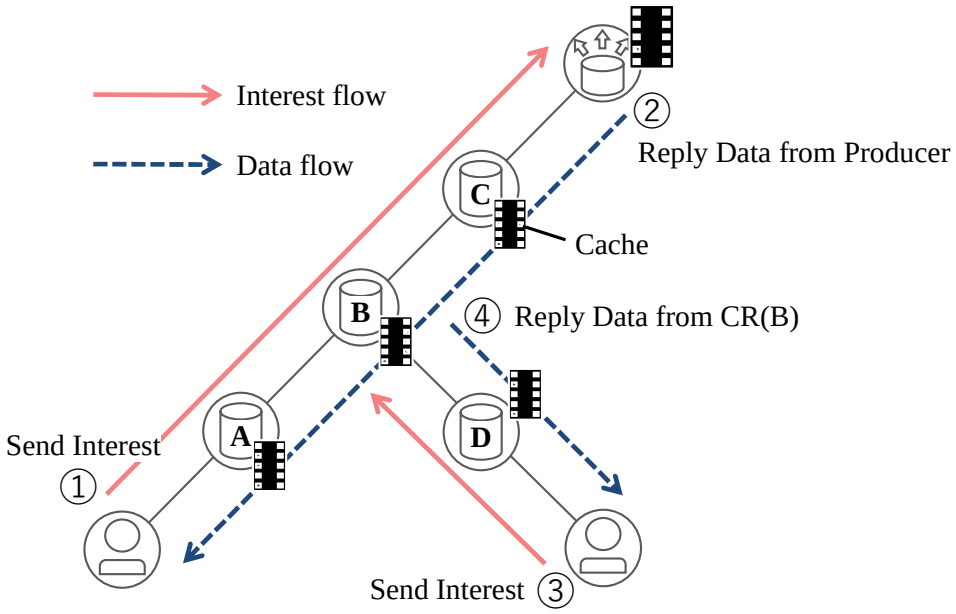


Figure 2.1: An overview of the ICN communication

layer (L2) with the resolution of challenges related to content distribution, without the need for additional functionalities such as edge servers or CDN servers. This marks a distinctive paradigm shift, and it is considered a technology that should be realized.

2.2 Operation of ICN

This section describes the operation of ICN. The design concept of ICN has existed for a while, and it has been derived into various types since then. Content-centric networking (CCN) [8] / Named data networking (NDN) [9], Pursuing a pub/sub internet (PURSUIT) [22], and Data-oriented network (DONA) [23] have been proposed are typical ICN architectures. I focus on a representative NDN environment in this study. Figure 2.1 shows the overview on the ICN communication. In NDN, a consumer requests content by sending an Interest packet that contains the name of the desired content. Note that an Interest packet is a request for a chunk of content called a Data packet. The CR that receives the Interest packet forwards it to a producer on the basis of the forwarding

information base (FIB) routing table. The producer then returns Data packets of requested content on the reverse path to the consumer. The CRs cache Data packets in their content store (CS) during forwarding, so they can return caches to the consumer instead of the producer if they store the requested Data. This in-network caching can satisfy the future requests of consumers, significantly reducing the network load and delivering content more efficiently.

2.3 Caching and routing schemes in ICN

However, the cache size of CR is significantly smaller than the huge amount of content worldwide, so it is naturally impossible to cache all content. In recent studies, the cache size of CR has been evaluated by assuming approximately 0.1–1.0% of the amount of total content. Therefore, an important issue to be addressed by ICN is to consider how to use these cashes, since they significantly affect the performance of content retrieval, and many efficient content- caching schemes have been proposed [10].

2.3.1 Simple caching scheme

The caching scheme is to consider what to cash and what to discard within a finite resource. The first step in these schemes was to resolve requests with a single CR. There have been three well-known cache-discard algorithm policies since long ago, such as LRU, LFU, and FIFO [24]. The purpose of these is to retain the content that is most likely to be used by consumers, i.e., the popular content that is frequently requested. However, a single CR does not take into account the caches on nearby CRs, which causes a cache redundancy problem in which only highly popular content is more replicated in the network, and thus cache utilization efficiency may be reduced. To solve this problem, several schemes have been proposed that organize the caches on the shortest path between consumer and producer. The major idea is to use a probabilistic factor when making cache decisions, such as RND [25] and UniCache [29]. In UniCache, for example, if there are four CRs on the shortest path and each CR caches content with a probability of 1/4, the content is approximately uniformly distributed among the CRs, thus preventing redundant caching. With these ideas, the cache redundancy problem

has been nearly solved. The next major discussion point is where and what to cache. ProbCache [26] controlled the cache location of the content by probability. The main idea of this scheme is that the probability of caching is adjusted for each content accounting with the distance to the producer. However, there is a limit to strictly control the placement of content by the probability of caching. In contrast, Leave Copy Down (LCD) [27], Move Copy Down (MCD) [28], and WAVE [29] have been proposed. These schemes take into account the popularity of the content, the centrality of the node, etc., and the CR explicitly moves the cache to the appropriate location. The key idea of these schemes is that each CR moves requested caches to downstream CRs. Namely, the CR caching the requested content sends its cache to the downstream CR, which can efficiently place popular content in the neighborhoods of consumers.

Consequently, these studies showed that it is important to consider the cache redundancy, the content popularity, and the placement of content in order to improve the cache efficiency. While the above schemes can achieve significant improvements, the amount of cache (even including caches on the shortest path) is still tiny compared to the huge amount of content worldwide. As a result, we believe that improvements in these ideas have nearly reached their limits.

2.3.2 Off-path caching scheme

Therefore, some schemes have been proposed to use the surrounding CRs instead of constraining to the shortest path. Such schemes that enable to search the cache outside of the shortest path by using additional FIBs are called off-path caching schemes, i.e., the above described schemes are called on-path cashing schemes. As the simplest off-path caching scheme, breadcrumb [30, 31] has been proposed. CRs maintain content delivery trail information and use it to directly forward requests. These schemes are on the basis of the idea that caches can be found through the interface that previously forwarded the content. Namely, requests are forwarded to the interface to be expected cache hit. This design concept has the problem that content delivery trail information becomes a false positive when content is discarded due to cache updates. This causes significant delays because the CR would forward the request on a large detour route. To improve accuracy regard to forwarding, some recent schemes use reinforcement

learning [32]. However, even with these schemes, the improvement in accuracy is still insufficient, so Scope-flooding [33], MuNCC [34], and SCAN [35] have been proposed. Scoped-Flooding gets caches from the nearest CRs by flooding requests that actively explore caches. In contrast, MuNCC and SCAN adequately maintain the routing table for nearby caches by exchanging cache information among CRs that passively update cache statuses.

2.4 Clustering scheme

Off-path caching extends the exploration range on the basis of an additional routing table, but simply combining it with the caching scheme described above does not achieve sufficient improvement because of redundant caches when considering between caches on shortest-paths. Therefore, routing and caching should be considered in an integrated manner, and the cache should be placed efficiently within the cache exploration range. In recent years, the exploration range has been replaced by the term "cluster", and clustering schemes [13, 14] (referred to as HRC and VDHTC hereafter, respectively), PoolCache [15], and HCC [16] have been proposed. These schemes group CRs into clusters in a domain and retain the main popular content in each cluster using a Hash-routing-like distributed caching manner [39]. The delivery latency can thus be controlled by cluster size, enabling consumers to retrieve content efficiently from the originating clusters. As a scheme similar to the aforementioned ones, the HCC [16] scheme has also been proposed. It centrally manages the distributed caches by a cluster header constructed in each cluster. The cluster header calculates the content popularity and importance of each node on the basis of information collected from the cluster, and then assigns the more popular content to the more important node to improve cache efficiency and delivery latency.

2.4.1 Issues with clustering schemes

This allows for the resolution of the majority of consumer content requests by utilizing a near cluster that allows for the retention of a large amount of popular content. However, I believe that we can achieve higher efficiency by focusing on two points, which are still open for discussion:

1. How to efficiently use surrounding clusters. The routing design of these clustering schemes has not allowed forwarding outside the cluster for request resolution. Specifically, the cache exploration range is restricted to the cluster boundaries, and the requests for non-cached content within the cluster are forwarded directly to the producer. This is the result by considering the risk that when using out-of-cluster to resolve requests, the paths tend to be larger due to mainly false positives, which may cause significant delays in content retrieval. However, although a cluster has a large cache size, it cannot cache the huge amount content worldwide, so the caches in neighboring clusters should also be made available. Therefore, routing that is not restricted by cluster boundaries is necessary.

2. How to adapt to changes in content demand trends. The design of these clustering schemes aims to improve cache utilization and reduce delivery latency by retaining a sufficient amount of the main popular content in the caches of a cluster. However, in a practical environment, the content demand trends, i.e., the amount of the main popular content, will change over time [17]. A too-small cluster size compared with the current amount of main popular content decreases cache utilization and causes delivery delays due to the delivery from producers, while a too-large cluster size increases cache utilization but may cause delivery delays due to long cache delivery. Therefore, I believe that the adequate cache distribution range should be determined in accordance with content popularity on the basis of such trade-off factors. Therefore, the cluster size needs to be determined depending on the situation.

3 Efficient content distribution and exploration scheme for clustering schemes

In this chapter, I discuss how to solve the issue 1. I propose an efficient distributed caching and exploration scheme for clustering schemes to address the issue 1. Furthermore, I discuss the effectiveness of the proposed scheme through simulation evaluations.

3.1 Introduction

I will first reorganize the issue 1 as described in Chapter 2. Various clustering schemes [13–16] have been proposed to reduce content delivery latency and improve cache efficiency.

HRC [13], VDHTC [14], and PoolCache [15] group several CRs into clusters in the network and apply Hash-routing into each cluster. These schemes enable to explore content from CRs using a hash function that maps content identifiers to each CR, i.e., the assignment location of the content, within a cluster. In particular, when a CR in a cluster receives a request, it calculates the hash value from the identifier of the received content and forwards it to the responsible CR in the cluster. If the content is not cached in the responsible CR, the request is then forwarded to the direction of the producer, in other words, the off-path routing is performed only in the originating cluster where the request is generated from the consumer. In HCC [16], the central management node, which manages the content assignment for each CR in the cluster, has the role of forwarding requests within the cluster. In particular, when the central management node receives a

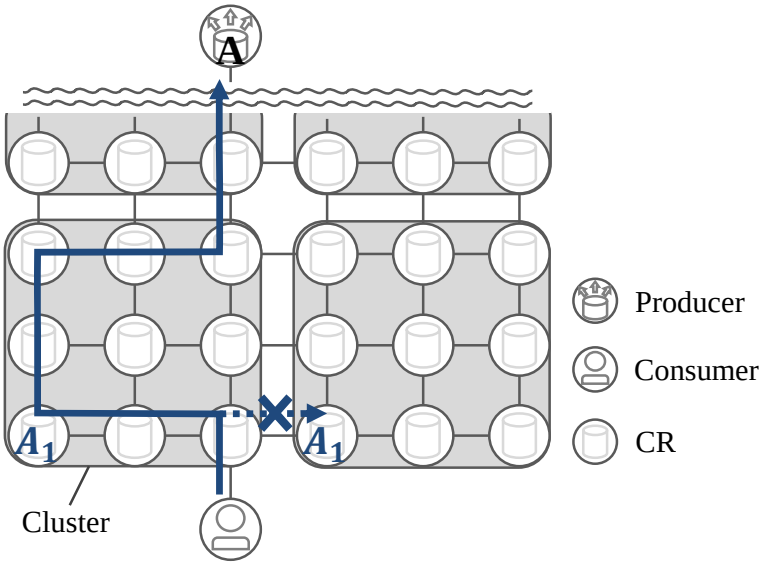


Figure 3.1: Inefficient content retrieval

request, it calculates the CRs in the cluster where the requested content may be cached and multicasts the requests in the direction of all CRs obtained from the calculation.

These approaches commonly cannot resolve the requests by using outside the clusters because they mainly perform intra-cluster routing to explore the contents. This design is the result by considering the risk that when using outside the clusters to resolve the requests, it may cause significant delays in content retrieval due to the paths for content exploration tend to be larger [13]. However, although a cluster has a large cache size, it cannot cache all content in the world, so the caches in neighboring clusters should also be available, as shown in Fig. 3.1. Therefore, routing approach that is not restricted by cluster boundaries is necessary.

I therefore propose an efficient distributed caching and exploration scheme for clustering schemes. The proposed scheme distributes contents uniformly to CRs in each cluster similarly to conventional clustering-based hash-routing. Moreover, it dynamically updates routing tables to enable consumers to retrieve distributed contents from anywhere regardless of cluster boundaries. I evaluate the effectiveness of the proposed scheme compared with conventional schemes through

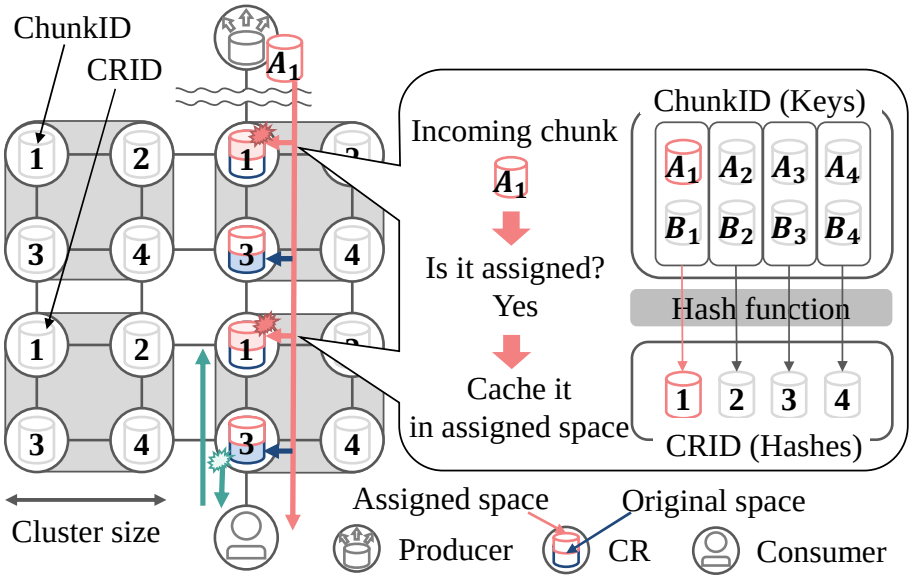


Figure 3.2: Distributed caching operation

simulations.

3.2 Proposed scheme

To improve content delivery latency and cache efficiency, I propose the cluster-based cache distribution and routing scheme. It groups CRs into clusters in a domain and retains the main popular content in each cluster using a distributed caching manner, enabling consumers to retrieve content from the originating clusters. Furthermore, it can also retrieve caches from closer CRs regardless of cluster boundaries by advertising cache information among CRs. In the following, I explain two functions of cluster-based distributed caching and advertisement-based routing.

3.2.1 Cluster-based Distributed Caching

The distributed caching approach uniformly distributes chunks of individual content to all CRs in each cluster, as shown in Fig. 3.2. This approach improves cache efficiency by avoiding cache duplication in the cluster, leading to more

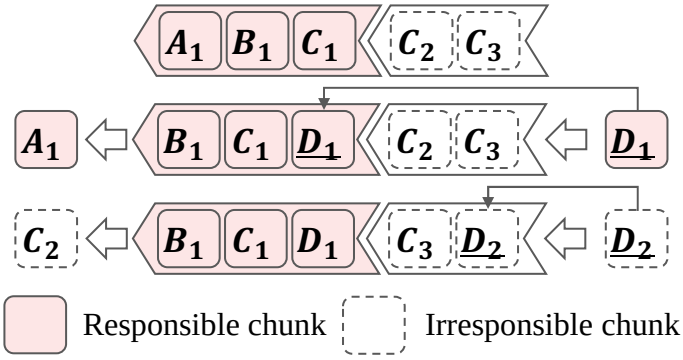


Figure 3.3: Cache replacement operation

cached content in it. Furthermore, transmission efficiency can also be improved by multi-path cache delivery from multiple CRs (i.e., load balancing).

As another aspect of the proposed scheme, I can expect a synergistic effect on content delivery latency by performing distributed caching while retaining the advantages of the original ICN's caching feature such as LCE, which uniformly caches received content during forwarding. This mechanism improves delivery latency by potentially retrieving requested chunks on the path to distributed caches, especially for popular content, as shown in Fig. 3.2 (green-colored arrow) where a consumer requests a chunk distributed cached in CR1 and retrieves the chunk from closer CR3 if cached.

Furthermore, the content placement is designed to maximize the benefits of the proposed routing scheme, which is not restricted by cluster boundaries. The conventional clustering schemes also assign contents with no duplication of caches within a cluster. However, when considering the range including the neighborhood clusters, the same contents may be assigned near the boundaries of each cluster, leading to redundant caches around consumers in the vicinity of these boundaries. In this case, content delivery latency may be longer due to unfairness of content retrieval by consumer's location, i.e., the consumer at the edge of a cluster is farther away from the caches than the consumer at the center of a cluster. Therefore, the proposed scheme uniformly distributes content with equal distance between the same content by separating clusters with the same size and the same content placement. As a result, contents in any CR does not duplicate with surrounding caches, and thus achieves fair content retrieval and high cache

utilization regardless of the consumer's location.

To uniformly distribute chunks in a cluster, this scheme partitions a domain into clusters of the same size and assigns unique identifiers (CRIDs) to each CR in advance, as shown in Fig. 3.2. To avoid cache redundancy among CRs in a cluster, the proposed scheme uses a hash function that maps chunk identifiers (ChunkIDs) to CRIDs. Specifically, when a CR receives a chunk, it caches the chunk with priority if the hash value calculated from the received chunk identifier matches its own CRID. To achieve this operation, the CS of each CR is divided into areas where responsible chunks are cached in an assigned space and other chunks are cached in the original space. In an example shown in Fig. 3.2, chunk A_1 requested by consumer is cached in the assigned space on CRs of CRID 1, which is the hash value of it, and in the original space on other CRs.

The Least Recently Used (LRU) cache replacement algorithm is used for both spaces on CS. Figure 3.3 illustrates an example cache replacement operation with three assigned spaces on CS. When a responsible chunk D_1 is received, it is cached in the assigned space and the oldest chunk A_1 in the space is discarded. After that, when an irresponsible chunk D_2 is received, it is cached in the original space and chunk C_2 is discarded.

Each cluster is square-shaped in this study. The cluster size is defined as the number of hops on one side (example in Fig. 3.2 is 2), which is a parameter that can control the cache placement range. This may affect the amount of content that can be placed. As another parameter, the ratio of the responsible chunk area and the original area on CS may affect the content redundancy and the possibility of retrieving chunks on the path to the responsible CR.

3.2.2 Advertisement-based routing for cluster-based caching

Even if caches are uniformly distributed within a cluster, consumers may not efficiently retrieve all chunks of the requested content from the originating cluster. This is because not all chunks will be cached due to the limitation of total cache capacity in a cluster, or there may be caches on closer CRs in neighboring clusters than those in the originating cluster. Therefore, requests should be forwarded to

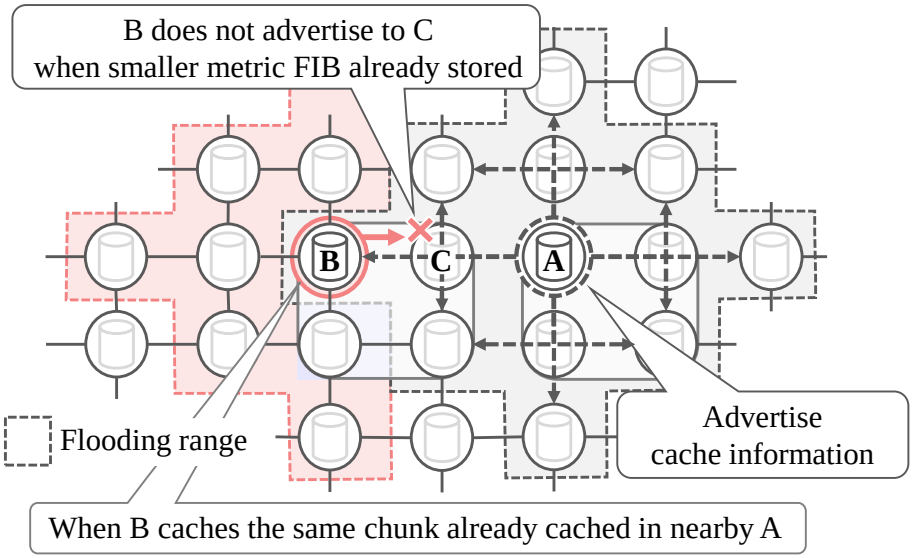


Figure 3.4: Cache information advertisement

the nearest caches even those not in the originating clusters regardless of cluster boundaries for efficient content delivery, so the advertisement-based routing approach is used, which forwards Interest packets to nearby caches on the basis of the advertised cache information.

Through this distributed caching manner, consumers can retrieve content from their own cluster where caches are uniformly distributed. However, it may not efficiently retrieve all chunks of the requested content from the originating cluster. Because not all chunks will be cached due to the limitation of total cache capacity in a cluster, or there may be caches on closer CRs in neighboring clusters than those in the originating cluster. Therefore, requests should be forwarded to the nearest caches even in not the originating clusters regardless of cluster boundaries for efficient content delivery, so the advertisement-based routing approach is used, which forwards Interest packets to nearby caches on the basis of the advertised cache information.

To achieve this behavior, each CR informs neighboring CRs of responsible cache status. Specifically, CRs that newly cache or discard responsible chunks advertise the cache information (newly cached/discarded) in the flooding manner regardless of cluster boundaries, as shown in Fig. 3.4. The CR receiving the

advertised packet updates its FIB entry with the received cache information. Therefore, CRs in the flooding range can forward Interest packets to the nearest CR caching responsible chunks.

Considering the overhead of this operation, the flooding range should be limited but would affect the content retrieval efficiency, which is defined as the flooding limit parameter (as shown in Fig. 3.4 is 2). This operation is performed only when responsible chunks are cached or discarded, thereby reducing the overhead compared with conventional schemes flooded for all cached chunks such as proposed in [36]. Moreover, to reduce the load caused by flooding, the proposed scheme simply discards and does not forward the flooding packets when it can be determined that neighboring CRs do not need to update their FIB. Let me explain this process using the example shown in Fig. 3.4. When CR A caches responsible chunks, it advertises its cache information to neighboring CRs (gray-colored range). After that, when CR B caches the same chunk, it can decide not to flood to CR C and advertises the cache information to neighboring CRs except it (red-colored range). This is because CR B has an FIB entry with metric of 2 hops for the chunk by advertised information from CR C and it indicates that CR C already has a valid metric of 1 hop that does not need updating. Namely, if the CRs already have FIB entries of plus 2 hops or fewer metrics than the flooding one, it does not need to advertise it in that direction. Note that this scheme increases overheads including cache information sharing and FIB entry increases to improve acquisition efficiency compared to on-path routing schemes as an inherent issue of off-path routing schemes. To resolve this issue (overheads caused by off-path extension), several solutions (e.g., a Bloom filter approach [34, 37, 38]) have been proposed, while I focus on reducing delivery latency by adjusting cache distribution range while considering only communication overheads caused by flooding in this study so that I will leave this issue for future work.

3.3 Simulation model

I evaluated the proposed scheme's effectiveness in retrieving content from nearby clusters/caches in a large domain environment using Network Simulator ns-3 ver. 3.30.1 [40] with the implementation of the proposed scheme. I used a simple

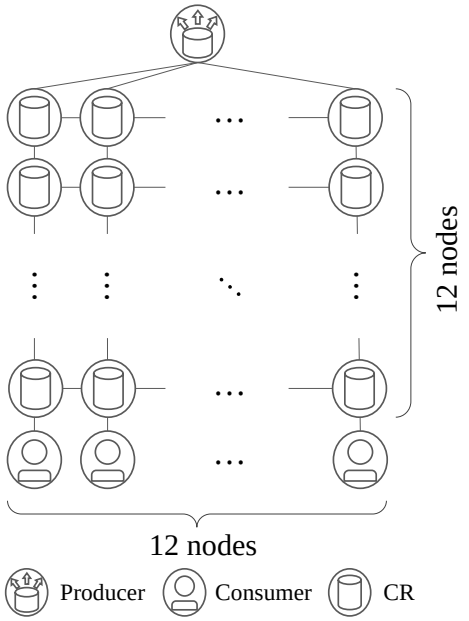


Figure 3.5: Simulation topology

grid topology with multiple paths to eliminate the effects of cluster shapes and content cache placement within clusters as shown in Fig. 3.5, which enable us to focus on the effect of distributed caching without strict cluster boundaries. This simulation topology formed an 12×12 grid of CRs with multiple paths. In this grid, 12 consumers were located on the lower side and one producer was located on the upper side. The parameters I used in the simulation are summarized in Table 1. The cluster is formed in squares, whose size is defined as the number of hops on one side. The ratio of the CS size on CR to the amount of content was set to approximately 1.5% on the basis of comparative papers [10, 13].

In this topology, each consumer sent Interest packets to request content toward the producer at normal distribution intervals with an average value of 0.3 seconds. The requested content was determined on the basis of the content popularity, in which P2P content was generally known to follow a Zipf-mandelbrot distribution [41]. In this distribution, the degree of bias depends on the parameters α and q. α is the skewness factor that controls the slope of a curve, while $q(\geq 0)$ is the plateau factor that decides the flatness of the curve. I define the top 15% of all content (All) as highly popular content (High). In this simulation, I gave

Table 3.1: Simulation parameters

Link bandwidths	1 [Gb/s]
Propagation delay time	5 [ms]
Amount of content	300
Content size	100 [chunk]
Chunk size	1,000 [Byte]
CS size on CR	500 [chunk]
Zipf α	0.4–2.0
Cluster size	2, 3, 4, 6, 12
Assigned space size on CS	50–500 [chunk]
FloodingTTL	2–16

q a fixed value of 5 and changed the content popularity with α for simplicity. Furthermore, I assumed no packet loss occurs so that I could focus on the fundamental characteristics when consumers retrieve content distributed within each cluster. The simulation was performed for 50 seconds, but 30–50 seconds was considered to avoid the effect of the transient period cached from empty to full in all the CSs.

In this simulation, I compared and evaluated the effectiveness of four representative schemes: LCE, Hash-routing (HR) [39], HRC [13], and proposed. The LRU cache replacement algorithm was used in each scheme. Note that, in HRC, if the requested content is not found on responsible CRs in a cluster, they forward Interest packets to a producer through the shortest path. This is because the requested content may not be found in any other clusters and thus exploding the content retrieval time should be avoided [13].

As mentioned in the previous Section 3.2, I considered three parameters (cluster size, ratio of assigned space and original space, and FloodingTTL) that affect performance, so I set each parameter in the following ranges and will clarify the characteristics. The cluster size was provided from 2, 3, 4, 6, to 12. The ratio of the assigned space is that divided by the CS size, which was provided from 10 to 100%. The FloodingTTL was provided from 2 to 16.

Furthermore, the average number of hops needed to retrieve content, the cache

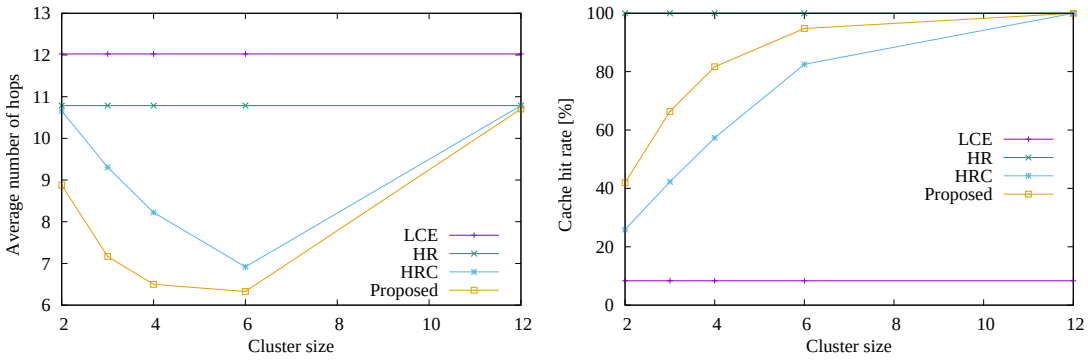
hit rate, and the advertisement rate were used as evaluation indices to discuss the effectiveness of the proposed scheme. The average number of hops that focused on content retrieval time was defined as the total number of hops when all consumers retrieved content, divided by the total number of consumer requests. The cache hit rate focused on cache efficiency and was the total number of cache hits on all CRs divided by the total number of requests for all consumers. I calculated it separately inside and outside the cluster to determine if consumers used caches outside clusters effectively. This was defined as the total number of cache hits out/in a cluster divided by the total number of requests for all consumers. The advertisement rate focused on overhead and was defined as the number of advertisement packets received per second in each CR.

3.4 Simulation results

In this section, first, I show the effectiveness of the proposed scheme compared with conventional schemes. Next, I investigate how parameters such as cluster sizes, ratios of assigned space, and FloodingTTL affect the proposed scheme. Finally, I clarify the scalability of the proposed scheme through various network scale evaluations.

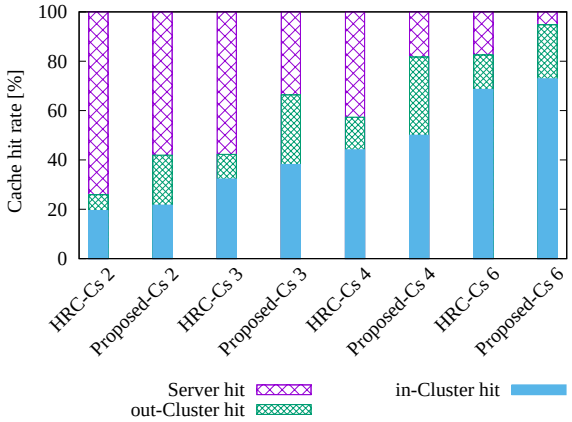
3.4.1 Effect of cluster size

In this section, first, I show simulation results and discuss the effectiveness of the proposed scheme compared with conventional schemes. Figure 3.6 shows the average number of hops, the cache hit rate, and the cache hit rate out/in a cluster when the cluster size (Cs) varies from 2, 3, 4, 6, to 12. Here, Zipf α was set to 0.8, the grid size was set to 12, the ratio of assigned space was set to 100%, and the FloodingTTL was set to 12 for the cluster size of 12 (w/o clustering) and 6 for other cluster sizes. The LCE scheme shows the worst performance among the other schemes because it causes duplicate caches on nearby CRs. The HR scheme attains better performance, especially cache efficiency, than the LCE scheme due to no duplicate cache occurrence, but the average number of hops, i.e., delivery latency is not good because the caches are distributed widely. The HRC scheme improves the performance compared with the HR scheme due to controlling the



(a) Average number of hops

(b) Cache hit rate

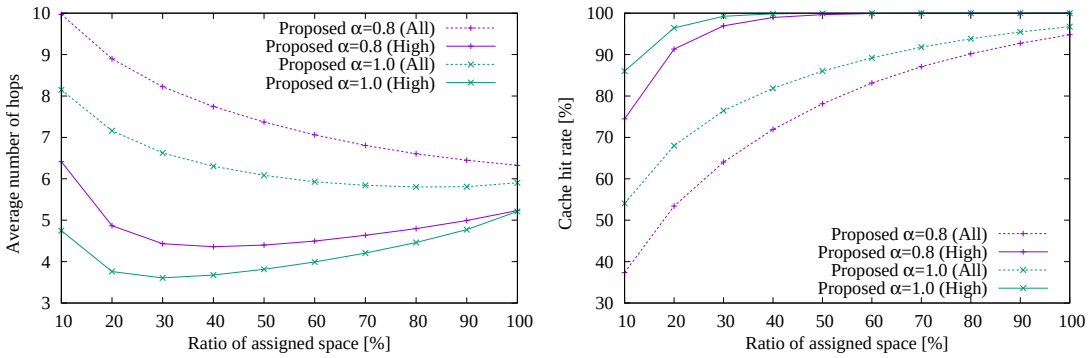


(c) Cache hit rate out/in cluster

Figure 3.6: Effect of cluster size

cache distribution range at the cost of a little cache efficiency. The proposed scheme further improves the performance compared with the HRC scheme. One reason for this improvement is that consumers can utilize nearby caches outside the cluster in addition to those inside one. Figure 3.6(c) shows that the proposed scheme significantly improves the cache hit rate outside the cluster. The other reason is that the proposed scheme can solve the false positive cache problem of Hash-routing [39]. the proposed scheme reduces the number of hops so that extra detours can be avoided from dynamic FIB updates.

Next, I focus on the effect of cluster sizes. the proposed scheme can improve cache efficiency with larger cluster sizes such as 12 (Fig. 3.6(b)) because larger



(a) Average number of hops

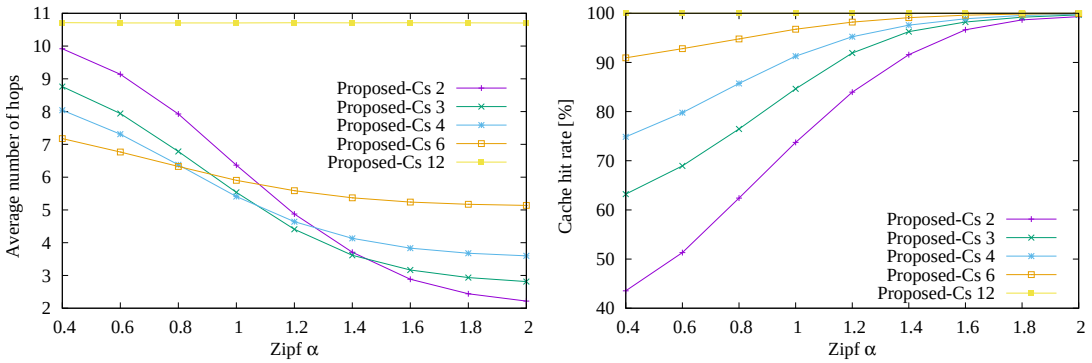
(b) Cache hit rate

Figure 3.7: Effect of ratio of assigned space

clusters can hold a greater variety of content. However, the content retrieval time increases (Fig. 3.6(a)) because the exploration range is extensive. However, the proposed scheme can reduce the content retrieval time in smaller cluster sizes such as 2, 3, 4 and 6 (Fig. 3.6(a)) because the exploration range is narrowed. However, if the cluster size is too small, the caches in the cluster may not satisfy requests alone, so content retrieval time eventually increases (Fig. 3.6(b)). Therefore, the adequate cluster size in this environment is 6. However, the cluster size must be set appropriately for the situation in the domain. A more detailed investigation is shared in Section 3.4.3.

3.4.2 Effect of ratio of assigned space

Next, I investigate the effect of the ratio of assigned space. Figure 3.7 shows the average number of hops and cache hit rate when the ratio of assigned space varies. Here, α was set to 0.8 or 1.0, the cluster size was set to 6 (optimal in this environment), the grid size was set to 12, and the FloodingTTL was set to 6. As Fig. 3.7(a) shows, when the assigned space is moderately small, the retrieval time for highly popular content (High) becomes shorter at the cost of increased retrieval time for all content (All). This tendency becomes especially apparent when α is large such as 1.0 and, thus, the requested content becomes more concentrated. This occurs because the original space running in the LRU cache retains the most popular content. Therefore, the original spaces of CRs on the path are utilized



(a) Average number of hops

(b) Cache hit rate

Figure 3.8: Effect of Zipf α

effectively when retrieving highly popular content. However, a smaller assigned space increases the overall retrieval time because it increases cache redundancy and decreases the cache hit rate as shown in Fig. 3.7(b). Namely, the ratio of assigned space has a trade-off between the retrieval efficiency of highly popular content and content overall.

Based on the above, the ratio of assigned space should be set on the basis of the requirements of service users and network administrators. For example, a large assigned space should be set if I want to develop a service that requires a variety of content on the domain. Conversely, a larger original space should be created if I want to mainly focus on delivering highly popular content.

3.4.3 Effect of Zipf α

In Section 3.4.1, I showed that a cluster size of 6 is suitable for this environment. However, the adequate cluster size in a practical network may vary on the basis of the situation in the domain. In this section, I investigate the effects of varying the distribution of content popularity. Figure 3.8 shows the average number of hops and the cache hit rate when α varies from 0.4 to 2.0. Here, the grid size was set to 12, the ratio of assigned space was set to 100%, and the FloodingTTL was set to 6. Figures 3.8(a) and 3.8(b) show how the cache hit rate tends to increase and the content retrieval time tends to decrease as α increases. However, the most important thing to note is that the adequate cluster shrinks as α grows. I can

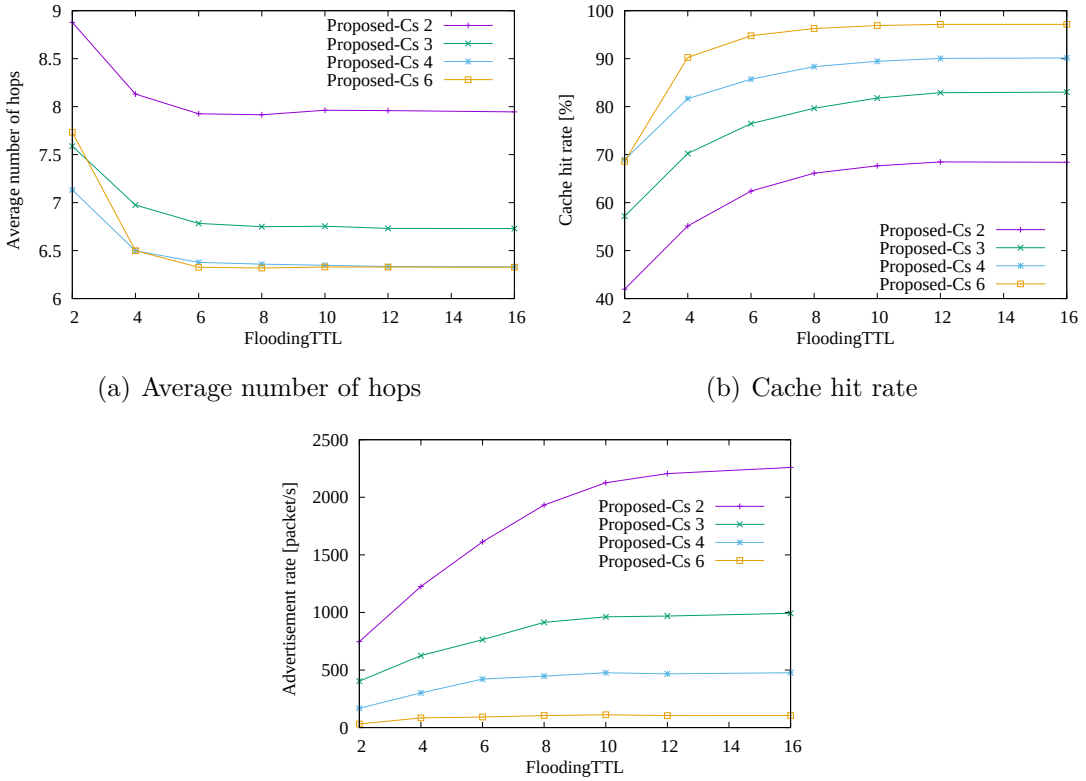
see that the adequate cluster size is 6 when α is less than 0.8, 4 for α of 0.8–1.0, 3 for α of 1.0–1.4, and 2 for α of 1.4 or larger, respectively, since these cluster sizes achieve the smallest number of hops for each content popularity. This is because the amount of main popular content decreases as α increases. Therefore, these contents can be placed in smaller cluster sizes. This allows consumers to retrieve content over a small range of exploration. Conversely, the amount of main popular content increases as α decreases, so I should maintain a high cache hit rate by increasing cluster sizes.

Therefore, in this environment, I can expect to maximize the improvement of content retrieval time by setting the cluster size to achieve a cache hit rate of about 80%. In a practical environment, the cluster size should be adjusted on the basis of an estimation of the degree of bias of the requested content.

3.4.4 Effect of FloodingTTL

I investigate the effects of FloodingTTL. Figure 3.9 shows the average number of hops, the cache hit rate, and the advertisement rate when the FloodingTTL varies. Here, α was set to 0.8, the grid size was set to 12, and the ratio of assigned space was set to 100%. These results show that the cache hit rate and the content retrieval time improve as FloodingTTL increases, although the overhead increases. This occurs because the CRs in a wider range can share cache information by a large FloodingTTL, so that they can forward Interest packets directly to the CR storing caches at the cost of increasing the number of advertisement packet transfers. Moreover, the advertisement rates gradually reach the upper limit regardless of cluster size as FloodingTTL increases. This occurs because larger FloodingTTL is more likely to find the same chunks, where CRs will not wastefully forward advertisement packets (recall the operation to reduce overhead described in Section 3.2.2).

However, I should note the disadvantages of setting FloodingTTL too large. the proposed graphs show that the degree of improvement of the cache hit rate and the average number of hops gradually decreases as the FloodingTTL increases. Therefore, a large FloodingTTL should not be set from a viewpoint of overheads. This occurs because the proposed cache placement scheme tends to make each cluster cache the same content (because each consumer tends to request the same



(a) Average number of hops

(b) Cache hit rate

(c) Advertisement rate

Figure 3.9: Effect of FloodingTTL

thing in this environment), so consumers are hard to find content that is not available in their own cluster from any other clusters even if the exploration range is expanded.

Finally, Fig. 3.9(c) shows that the proposed scheme with the cluster size of 6 achieves the lowest advertisement rate, i.e., overhead. This is because the adequate cluster size can sufficiently cache the main popular content in each cluster as well as cache updates rarely occur. Therefore, I should set the cluster size to 6 taking into account overhead (affected by FloodingTTL) in addition to the discussion in the previous Section 3.4.3, although the cluster size of 4 with sufficient FloodingTTL attains the same performance in terms of content retrieval time.

Based on the above, in this environment, I recommend setting the FloodingTTL

to about 4 or 6, considering the degree of improvement against the overhead. However, in a practical environment, the caches in each cluster may differ depending on the request locality of each consumer. I will investigate this point in the future.

3.5 Conclusion

A network clustering schemes have been proposed to improve cache efficiency and delivery latency in ICN. These schemes group CRs into clusters in the network and apply intra-cluster routing into each cluster. However, it may not fully exploit distributed network caches because the content exploration range is strictly bounded within a corresponding cluster. Therefore, I proposed an efficient distributed caching and exploration scheme for cluster-based ICN. the proposed scheme distributes content to each cluster in a distributed caching manner. In addition, it dynamically updates the FIB on the basis of collaboration among CRs across the cluster boundary. Through simulations, the proposed scheme demonstrated more efficient caching and faster retrieval than conventional schemes by using nearby clusters/caches.

4 Popularity-aware dynamic clustering scheme

In this chapter, I discuss how to solve the issue 2. I propose the extended scheme of the proposed scheme in Chapter 3 to address the issue 2. This extension involves dynamically adjusting the cluster size according to fluctuations in content demand trends. Furthermore, I discuss the effectiveness of the proposed scheme through simulation evaluations.

4.1 Introduction

I will reorganize the issue 2 as described in Chapter 2. To reduce the content delivery latency and improve the cache efficiency, clustering schemes have been proposed [13–16, 18]. As aforementioned in Chapter 2, these schemes group CRs into clusters in a domain. Each cluster aims to avoid cache duplication among CRs and to cache a sufficient amount of the main popular content within it. As a result, these schemes enable consumers to retrieve content efficiently from the originating cluster. However, the amount of content that can be cached in the cluster depends on the cluster size, i.e., cache distribution range. In other words, a smaller cluster size is insufficient to reduce delivery latency since it cannot cache the main popular contents sufficiently in the cluster. I therefore believe that it is necessary to determine the adequate cluster size in accordance with content popularity on the basis of the following trade-off factors. A too-small cluster size against the amount of main popular content will not retain sufficient caches, so it decreases cache utilization and causes delivery delays due to the delivery from producers. A too-large cluster size can satisfy most requests within the cluster but causes delivery delays due to the delivery from widely distributed caches. In

a practical environment, the distribution of content popularity, i.e., the amount of main popular content, will change over time [17], so it is necessary to determine the cluster size adequately depending on the situation.

Several studies have also focused on similar issues related to content popularity with static clustering schemes that do not adjust cluster size in real time. In HRC [13], a bin-packing algorithm have been proposed to determine the location of content in a cluster. The algorithm outputs content placement that assigns even load to each node on the basis of previously studied network characteristics, taking into account node importance, content popularity, and tendency of requests. In addition, it places each content into CRs that provide the most average latency gain, starting with the most popular content. As a result, it not only shows an efficiency improvement in content delivery latency while distributing the link load, but also nearly adapts to fluctuations in popularity. HCC [16] determines content placement by calculating a probability matrix on the basis of node importance and content popularity. The design of probability matrix is that more important nodes should cache more popular contents and less important nodes should cache more less popular contents. In addition, the content placement is periodically calculated and updated to account for fluctuations in content popularity over time.

These approaches address the problem of fluctuations in content popularity within static clusters. However, although it is necessary to calculate in real time for adapting the fluctuations of content popularity, the volume of these calculations faces a challenge in terms of feasibility. Moreover, when the static clusters are too small compared to the amount of main popular content, the above problem may not be solved, even if the placement of content within the clusters is efficient.

I therefore propose a dynamic clustering scheme to adjust the cluster size, in accordance with the change in content popularity, considering cache utilization and delivery latency. The proposed scheme controls the cluster size effectively using a simple threshold-based algorithm on the basis of the number of cache updates on CRs in a cluster. Moreover, I evaluate the effectiveness of the proposed scheme compared with conventional schemes through simulation in a situation where content popularity changes.

4.2 Proposed scheme

As previously mentioned, the cluster size, i.e., cache distribution range, should be adequately determined in accordance with content popularity. In a practical environment, the distribution of content popularity changes over time, so it is necessary to determine the cluster size depending on the situation. I therefore propose a dynamic clustering scheme to adjust the cluster size in accordance with the change in content popularity, considering cache utilization and delivery latency. The proposed scheme controls the cluster size effectively using a simple threshold-based algorithm on the basis of the number of cache updates in the cluster.

To discuss the adequate cluster size, I focus on the frequency of cache updates in a cluster. This is because this metric is useful to estimate whether the current cluster size is suitable to cache the main popular content. When the frequency of cache updates is high, it indicates that caches are updated by incoming data packets from outside the cluster. Namely, requested content cannot be retrieved inside the cluster as well as the cluster size is too small. A low frequency of cache updates indicates that caches are not updated since requested content can be retrieved inside the cluster. Namely, the cluster size may be decreased to reduce delivery latency. Thus, I consider that the frequency of cache updates in a cluster would fall into a certain range with the appropriate cluster size.

From the aforementioned strategy, the proposed scheme adjusts the cluster size using a simple threshold-based algorithm on the basis of the frequency of cache updates. Specifically, it uses the number of cache updates in a cluster as a metric, and decreases/increases the cluster size when the metric falls below or exceeds lower/upper thresholds. Figure 4.1 explains how the proposed scheme migrates to the adequate cluster size in accordance with the change in content popularity. Let us consider a t-second scenario when the content popularity will disperse after x seconds, and then heavily concentrate after y seconds. In phase 1 until x seconds, I assume that each cluster, which represents the domain divided into four parts, can store most of the popular content, so the frequency of cache updates fits between the upper and lower thresholds. Namely, the current cluster size is adequate. In phase 2 from x to y seconds when the content popularity disperses, the frequency of cache updates increases and exceeds the upper threshold because

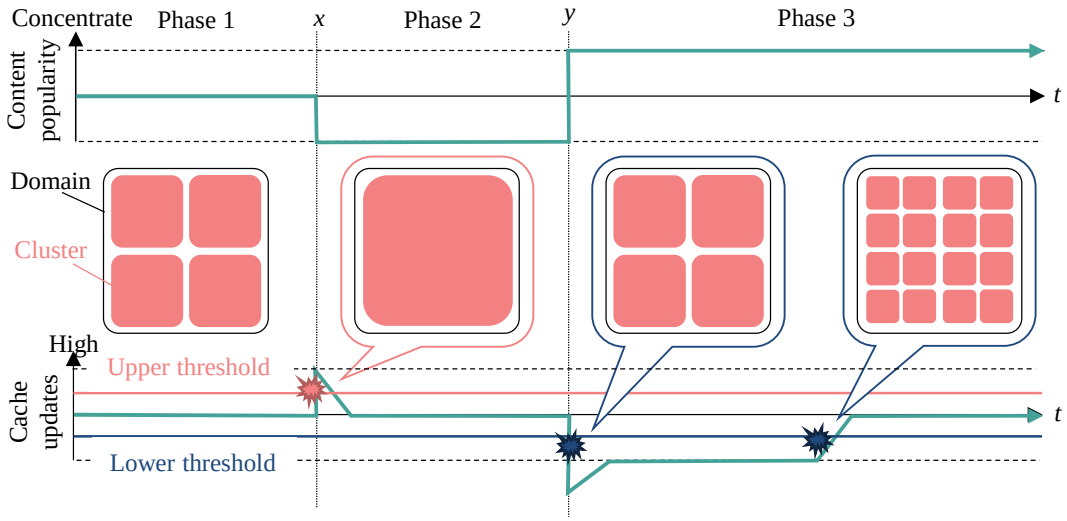


Figure 4.1: Operation of dynamic clustering

the current cluster size cannot retain the popular content sufficiently. Therefore, the cluster size is increased by one level to store them, and therefore the frequency of cache updates decreases and falls within the upper and lower thresholds. In phase 3 after y seconds when the content popularity is heavily concentrated, the cache update frequency decreases and falls below the lower threshold because the current large cluster size has exceeded the sufficient cache capacity compared with the amount of main popular content. Therefore, it attempts to improve the delivery latency by decreasing the cluster size by one level. However, this cluster size still has an excessive cache capacity, so the frequency of cache updates remains below the lower threshold. Therefore, the cluster size is decreased by one more level, and therefore the frequency of cache updates increases and falls within the upper and lower thresholds. Through these procedures, the cluster size can be migrated to the adequate cluster size in accordance with the change in content popularity.

To achieve this function, I assume that a controller is located in a domain and each CR notifies the controller with the number of cache updates. The controller calculates the total number of cache updates separately in each cluster by the information received from each CR. When at least one of the calculated values falls below or exceeds lower/upper thresholds, it reassigns a new CRID

Table 4.1: Simulation parameters

Link bandwidth	1 [Gb/s]
Propagation delay time	5 [ms]
Amount of Contents	128
Content size	64 [chunk]
Chunk size	1000 [Byte]
CS size on CR	128 [chunk]
Zipf α	0.6, 1.0, 1.4, 1.8
Cluster size	2, 3, 4, 6, 12
Upper threshold	70–280
Lower threshold	10–150
Reclustering interval	1–24 [s]

and hash function to each CR to decrease/increase cluster size. The cluster size is not changed for a certain period, which is defined as the reclustering interval parameter, immediately after reclustering to mitigate the effect of the heavy fluctuation of cache updates. I believe that such information sharing between the controller and CRs can be achieved by a mechanism like software defined networking (SDN) and the detailed design of the scheme will be left as future work.

4.3 Simulation model

I evaluated the proposed scheme focusing on the effectiveness of retrieving content from nearby clusters/caches in a large domain environment where content popularity changes through simulations using Network Simulator ns-3 ver. 3.30.1 [40] with the implementation of the proposed scheme. I used a simple grid topology with multiple paths to eliminate the effects of cluster shape and content cache placement within clusters as shown in Fig. 4.2 to enable us to focus on the essential effect of dynamically changing cluster size. One producer and 12 consumers were located on the upper and lower sides of the grid (12 x 12) of CRs, respec-

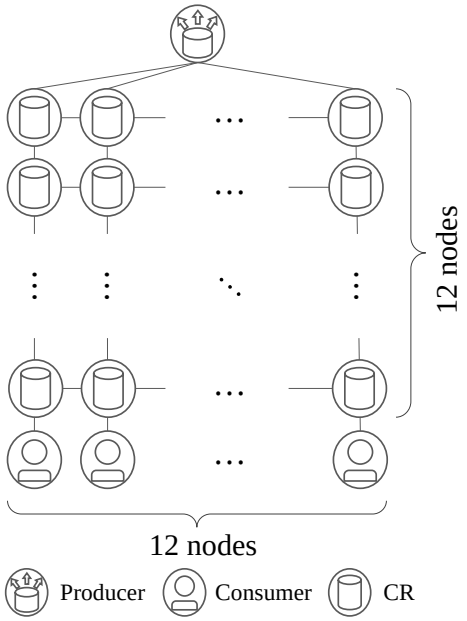


Figure 4.2: Simulation topology

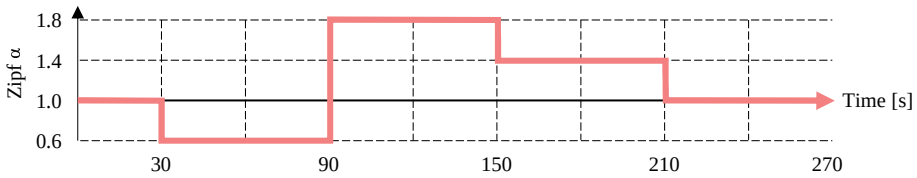


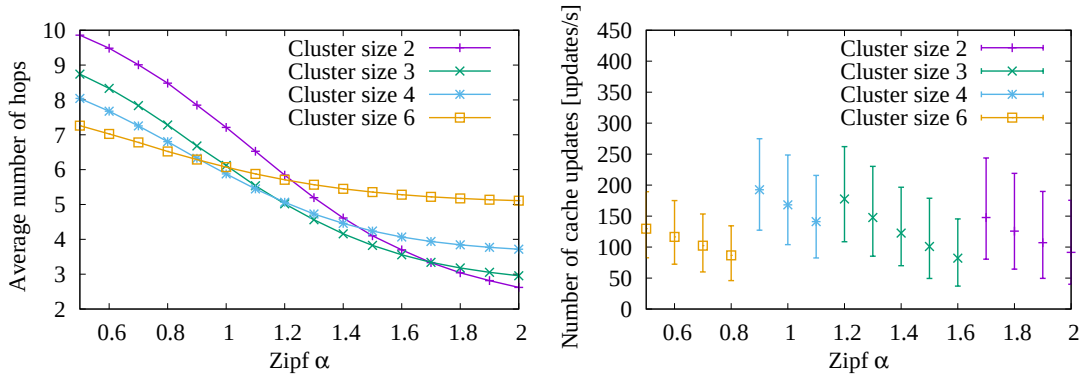
Figure 4.3: Fluctuation of zipf α

tively. The parameters used in the simulation are summarized in Table 4.1. The ratio of the CS size on CR to the amount of content was set to approximately 1.5% on the basis of comparative papers [10, 13]. The flooding limit was set to 6, which was the best value in terms of cost performance between overhead and efficiency in a preliminary evaluation. As mentioned before, the proposed scheme needs to share information among CRs via the controller, which can be achieved by a number of mechanisms like SDN, and I ignore its effect in this simulation since the exchange of shared information is very infrequent and small compared with data delivery. Each CR notifies the controller of the number of cache updates at 1 second intervals.

Each consumer sent interest packets to request content toward the producer at

normal distribution intervals with an average value of 0.3 seconds. The requested content was determined on the basis of the content popularity, in which P2P content was generally known to follow a Zipf-mandelbrot distribution [41], and I gave q a fixed value of 5 and changed the content popularity with α to avoid the complexity of the discussion. Furthermore, I assumed no packet loss occurs so we can focus on the fundamental characteristics of the dynamic clustering approach. The simulation was performed for 270 seconds. I set the Zipf parameter α to 1.0 at the start of the simulation as shown in Fig. 4.3. α changed to 0.6 at 30 seconds after the simulation started, in which a wider range of content is requested, to 1.8 at 90 seconds, to concentrate on the requested content, and after that, it decreases by 0.4 every 60 seconds back to 1.0.

In this simulation, I compared and evaluated the effectiveness of five representative schemes: LCE, HR [39], HRC [13], Static [18], and the proposed scheme in this chapter (Dynamic). Note that the HRC scheme uses the k-split algorithm with the number of hops as similarity metrics for clustering and forms k clusters, and the Static scheme is the proposed scheme in Chapter 3. Furthermore, the average number of hops needed to retrieve content, cache hit rate, and advertisement rate were used as evaluation indices to discuss the effectiveness of the proposed scheme. The average number of hops focused on content retrieval time, which was defined as the total number of hops during the time when all consumers retrieved content divided by the total number of requests for all consumers. The cache hit rate focused on cache efficiency, which was defined as the total number of cache hits on all CRs divided by the total number of requests for all consumers. The advertisement rate focused on communication overhead, which was defined as the amount of advertisement packets divided by the total amount of traffic. In this study, I assumed the average name length is 30 bytes, and the size of the advertisement packet which includes the content name, the flooding limit, and the flag bit that indicates the cache information (newly cached/discarded), is the same as the Interest packet.



(a) Average number of hops

(b) Number of cache updates

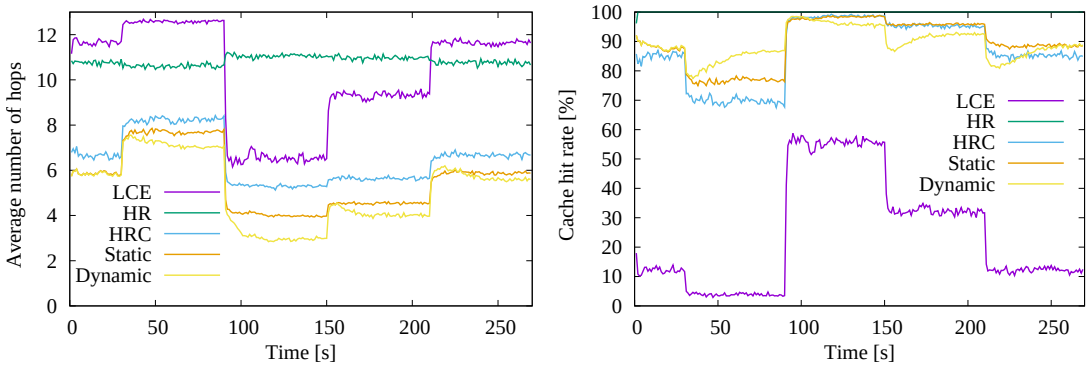
Figure 4.4: Estimation of adequate thresholds

4.4 Simulation results

In this section, I first show the effectiveness of the proposed scheme compared with the conventional schemes. Then, I investigate how each parameter including the lower/upper thresholds and reclustering interval affects the proposed scheme. Finally, I investigate the effect of the change interval of Zipf α and network topology to reveal the environmental tolerance and practicality of the proposed scheme.

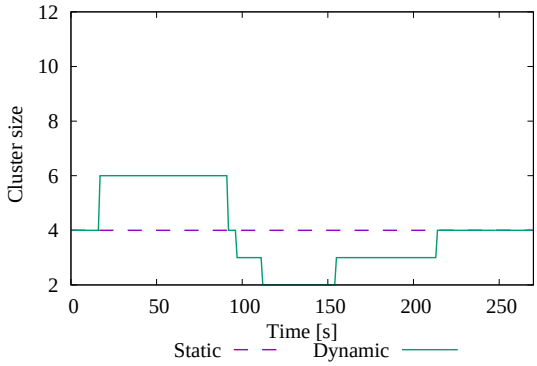
4.4.1 Evaluation of Effectiveness on the basis of Estimation of Adequate Thresholds

In this section, I first discuss the basis for determining threshold values of the proposed scheme through quantitative evaluations and estimate the effective lower/upper threshold values, which is a key point of the proposed scheme. As mentioned in Section 4.2, given an adequate cluster size, the number of cache updates in the cluster falls into a certain range. I believe that the adequate cluster size can be determined in accordance with the distribution of content popularity. Figure 4.4 shows the average number of hops and cache updates in the cluster when α varies from 0.5 to 2.0. From Fig. 4.4(a), we can see that the adequate cluster size is 6 when α is less than 0.9, 4 for α of 1.0–1.1, 3 for α of 1.2–1.6, and



(a) Average number of hops

(b) Cache hit rate



(c) Cluster size

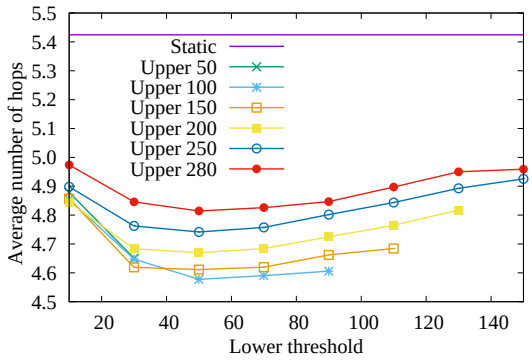
Figure 4.5: Effectiveness of the proposed scheme

2 for α of 1.7 or larger, respectively, since these cluster size achieve the smallest number of hops for each content popularity. Correspondingly, the number of cache updates in the cluster falls into a certain range when the adequate cluster size is given as shown in Fig. 4.4(b). Specifically, it is approximately 50 or more for the adequate cluster size of 6 ($\alpha = 0.9$ or less), 90–280 for the size of 4 ($\alpha =$ 1.0–1.1), 50–270 for the size of 3 ($\alpha =$ 1.2–1.6), and 250 or less for the size of 2 ($\alpha = 1.7$ or above), respectively. From the aforementioned results, if the number of cache updates in the cluster is approximately 50 and more or 280 and less, the given cluster size will be adequate. Namely, the lower/upper threshold values can be set on the basis of the number of cache updates.

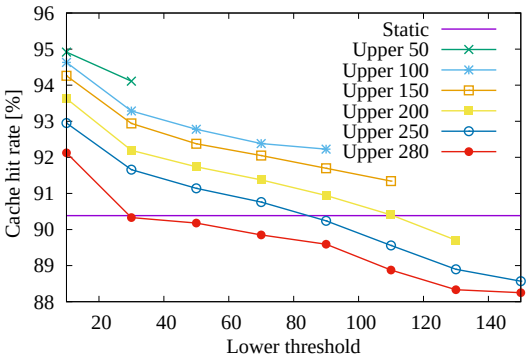
On the basis of the aforementioned discussion, I now show the simulation results and discuss the effectiveness of the proposed scheme as compared with the

conventional schemes. Here, the lower/upper threshold values were set to 50/280, the reclustering interval was set to 3 seconds, and the initial cluster size of HRC, Static, and proposed (Dynamic) schemes was set to 4, which was the appropriate value for an α of 1.0 at the start of the simulation. Figure 4.5 shows the average number of hops, cache hit rate, and cluster size as a function of time. From Figs. 4.5(a) and (b), the LCE scheme shows the worst performance among the other schemes because it causes duplicate caches on nearby CRs. The HR scheme improves the performance, especially cache efficiency, compared with the LCE scheme due to no duplicate cache occurrences, but the average number of hops, i.e., delivery latency is not good because the caches are distributed widely. The HRC scheme improves the performance compared with the HR scheme due to controlling the cache distribution range at the cost of a little cache efficiency. The Static scheme using advertisement-based routing improves the performance compared with the HRC scheme due to the avoidance of detour routing caused by the false-positive problem with the HR scheme as well as the effect of retrieving nearby caches regardless of cluster boundaries. The proposed (Dynamic) scheme further improves the delivery latency while maintaining the cache hit rate compared with the Static scheme in almost all ranges of time because it adjusts the cluster size to an adequate value.

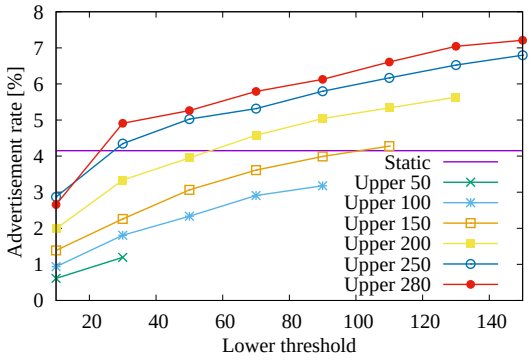
Next, let us take a look at adjusting the cluster size of the Dynamic scheme focusing on three periods where the content popularity changes. First, in the period of 30–90 seconds, a wider range of content becomes to be requested, so that the cluster size is adjusted to a larger value (it is 6, which is an adequate value when $\alpha = 0.6$ (Fig. 4.4(a))) due to the high frequency of cache updates as shown in Fig. 4.5(c). It improves the cache hit rate as well as delivery latency, although it takes time to distribute new caches in the cluster. Second, in the period of 90–150 seconds, the requested content becomes to be concentrated, so that the cluster size is adjusted to a smaller value (it is 2, which is an adequate value when $\alpha = 1.8$) due to the low frequency of cache updates. It improves the delivery latency, although it takes time to discard unnecessary caches from the cluster, and comes at the cost of a slight decrease in cache hit rate. Finally, in the period of 150–270 seconds, similar to 30–90 seconds the requested content becomes to be a wider range gradually, so that the cluster size is adjusted to



(a) Average number of hops



(b) Cache hit rate



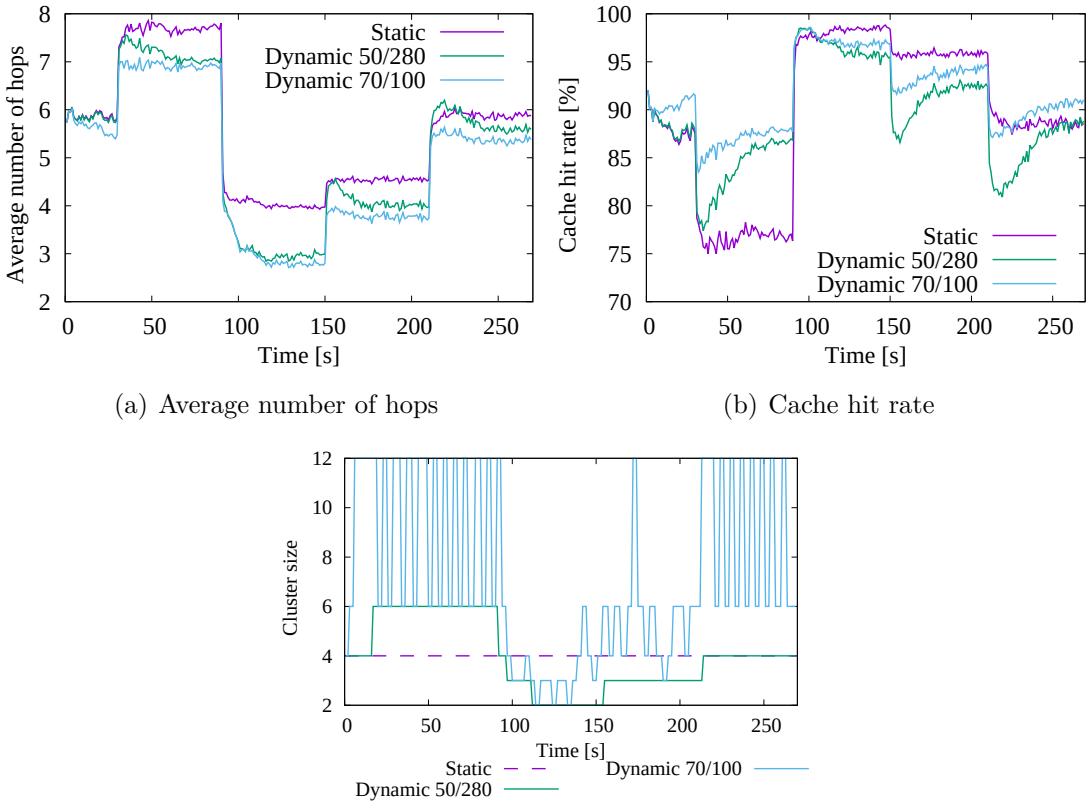
(c) Advertisement rate

Figure 4.6: Effect of thresholds

larger values (they are 3 and 4, which are adequate values when $\alpha = 1.4$ and 1.0, respectively). It improves delivery latency while maintaining a high cache hit rate. This adjustment of cluster size is performed by searching for the cluster size that keeps the number of cache updates in the range of 50 to 280. Therefore, the proposed scheme can adapt effectively to the environment where content popularity changes.

4.4.2 Effect of Thresholds

Next, I investigate the effect of the thresholds. Figures 4.6(a), (b), and (c) show the average number of hops, cache hit rate, and advertisement rate, respectively, when the lower/upper thresholds vary. Here, the reclustering interval was set to



(a) Average number of hops

(b) Cache hit rate

(c) Cluster size

Figure 4.7: Estimated and adequate thresholds

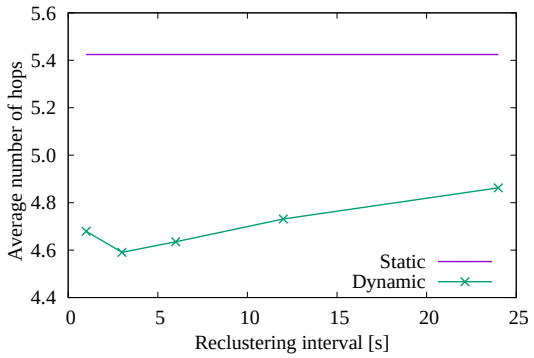
3 seconds. Figures 4.6(a) and (b) indicate that the upper and lower threshold values should be set to an appropriate range (neither too large nor too small) to reduce delivery latency and maintain the high cache hit rate. When the upper threshold value is too large, it is difficult to migrate to a larger cluster size despite high frequent cache updates. As a result, it worsens cache efficiency as well as delivery latency. When the upper threshold value is too small, it is easy to migrate to a larger cluster size despite low frequent cache updates. As a result, it improves cache efficiency but increases delivery latency because the caches are widely distributed. The lower threshold observes a similar trend. Consequently, the adequate threshold values should be determined on the basis of the delivery latency and cache hit rate considering these trade-offs. The adequate lower/upper thresholds are 70/100 in this simulation environment, which achieves

the lowest number of hops (Fig. 4.6(a)) and the high cache efficiency (Fig. 4.6(b)). Furthermore, the adequate cluster size does not cause frequent cache updates and reduces the flooding of advertisement packets for dynamic FIB updates, so the proposed (Dynamic) scheme with adequate thresholds also improves the advertisement rate, i.e., communication overhead, to approximately 2% of the total amount of traffic (Fig. 4.6(c)).

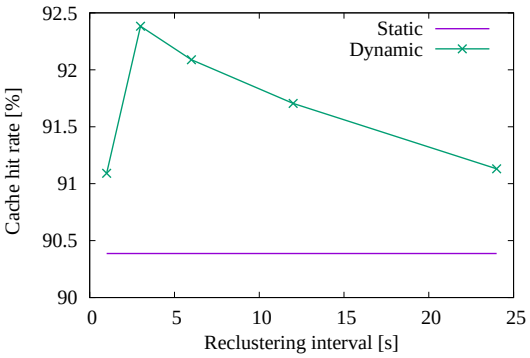
Here, it is noted that the estimated threshold values and adequate ones are largely different. This indicates that it should aggressively migrate to various sizes of clusters with the setting of larger/smaller lower/upper threshold values to maintain cache hit rates in the environment where the content popularity changes significantly. Figure 4.7 shows the average number of hops, cache hit rate, and change of cluster size in the Static scheme and proposed (Dynamic) scheme with the estimated (50/280) and adequate (70/100) threshold values. Figure 4.7(c) clearly shows that the proposed scheme with adequate thresholds can more frequently migrate closer to the appropriate cluster size than that with estimated thresholds. Moreover, Fig. 4.7(a) and (b) show that such migration quickly improves the delivery latency and cache hit rate when the content popularity changes. Consequently, although the proposed scheme with estimated thresholds achieves good performance, it can be further improved by setting adequate thresholds on the basis of the aforementioned trade-offs as well as detecting sensitive changes in content popularity to quickly adjust the cluster size with appropriate cache distribution. However, the adequate threshold values may need to be adjusted dynamically in accordance with network conditions (the topology, frequency of requests, etc.), which will be tackled in future work.

4.4.3 Effect of Reclustering Intervals

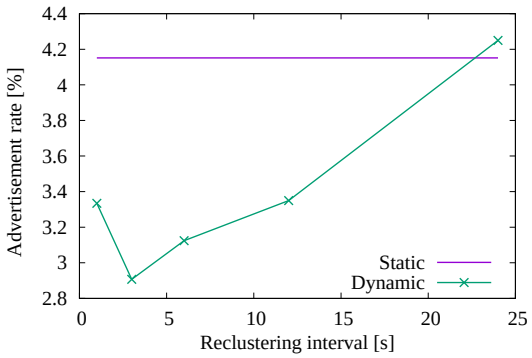
I investigate the effect of reclustering intervals. Figure 4.8 shows the average number of hops, cache hit rate, and advertisement rate when the reclustering interval varies. Here, the lower/upper thresholds were set to 70/100 (adequate values in this environment). Figure 4.8(a) shows that shorter reclustering intervals improve delivery latency except for too-short ones. This is because the shorter intervals can quickly migrate to the adequate cluster size and improve cache hit rates as shown in Fig. 4.8(b). However, too-short intervals inhibit mi-



(a) Average number of hops



(b) Cache hit rate



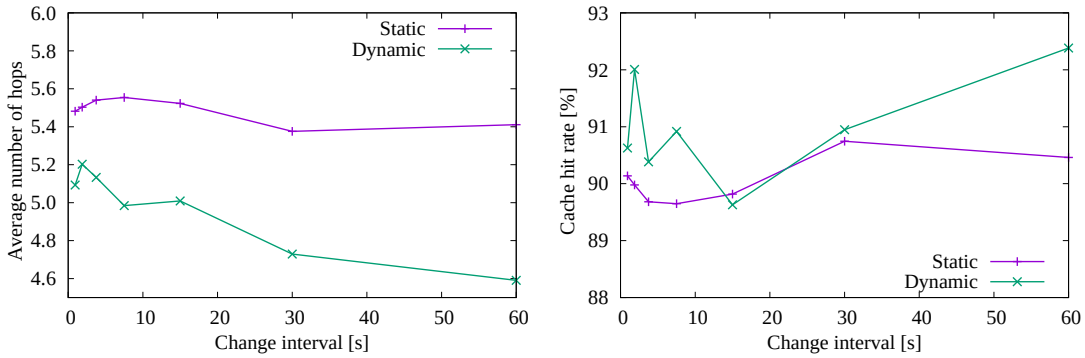
(c) Advertisement rate

Figure 4.8: Effect of reclustering intervals

gration to the adequate cluster size due to heavy cache updates immediately after reclustering. In addition, Fig. 4.8(c) shows that shorter reclustering intervals improve the overhead. This is because unnecessary cache updates are reduced by quickly migrating to the adequate cluster size. Consequently, the reclustering interval should be set to an adequately short value, which is 3 seconds in this environment.

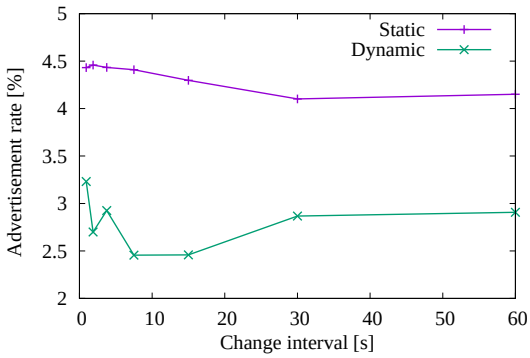
4.4.4 Effect of Change Intervals of Zipf

I investigate the effect of change intervals of Zipf α to show the environmental tolerance. For example, when the change intervals are set to 20 seconds, 30 seconds after the simulation starts with Zipf α of 1.0 and a cluster size of 4,



(a) Average number of hops

(b) Cache hit rate



(c) Advertisement rate

Figure 4.9: Effect of change intervals of Zipf α

Zipf α sequentially changes to 0.6, 1.4, 1.8, and 1.0 every 20 seconds, and these changes are repeated for 240 seconds (until the end of simulation). Figure 4.9 shows the average number of hops, cache hit rate, and advertisement rate when the change intervals of Zipf α vary. Here, the thresholds of lower/upper were set to 70/100, and the reclustering interval was set to 3 seconds (adequate values in this environment). From Fig. 4.9, the proposed (Dynamic) scheme always improves the delivery latency, cache hit rate, and overhead compared with the Static scheme in a wide range of change intervals. This is because the proposed scheme can adapt cluster sizes smoothly to environments where the content popularity changes frequently.

4.5 Conclusion

A network clustering schemes have been proposed to improve cache efficiency and delivery latency in ICN. The design of these schemes aims to improve cache utilization and reduce delivery latency by retaining a sufficient amount of the main popular content in the caches of a cluster. However, in a practical environment, the content demand trends, i.e., the amount of main popular content, will change over time. In other words, if cluster size is static such as these schemes, it may be impossible to cache enough content in a cluster to reduce delivery delays due to fluctuations in the amount of content. I therefore proposed a dynamic clustering scheme to adjust the cluster size in accordance with the change in content popularity. The proposed scheme adjusts the cluster size effectively using a simple threshold-based algorithm on the basis of the number of cache updates in the cluster. Simulation evaluations have indicated that the proposed scheme can reduce the delivery latency while consistently maintaining a high cache hit rate in a large domain environment where content popularity changes.

5 Evaluation of clustering schemes in practical environment

In this chapter, I delve into the adaptability of the proposed scheme in practical networks. I propose a clustering scheme for practical networks, and discuss the adaptability of the proposed scheme in actual networks through simulation evaluations.

5.1 Introduction

Throughout Chapters 3 and 4, I have presented the proposed schemes and their effectiveness in addressing issues 1 and 2. However, the design and evaluation of the proposed schemes were on the basis of the shape of a simple grid topology. When considering to apply the proposed schemes to practical topologies with complex shapes, it is difficult to partition the network into square clusters as in the proposed scheme. Therefore, it is necessary not only to find ways to apply the proposed scheme to practical topologies but also to evaluate how the performance and characteristics may depend on the shape of the topology/cluster.

In the clustering schemes such as HRC [13] and VDHTC [14], the network clustering approaches for practical topology are performed offline by precomputing well-known clustering algorithms such as k-split, k-means, or k-medoids. These algorithms aim to partition a domain network consisting of V nodes into K clusters that satisfy any given objective function using similarity metrics. These similarity metrics can be flexibly defined by network administrators, considering not only simple hop count or delay between node to node but also factors like

content popularity, bandwidth, and cache size. This flexibility enables the adjustment of cluster shapes to achieve the objective. In contrast, HCC [16] clusters the practical topology using a unique clustering approach on the basis of the Waited-based clustering algorithm. In this approach, any K central management nodes selected from within the network perform the online clustering. The selection is on the basis of the consideration of node degree, average delay, or hop count in the network/cluster, choosing the node that maximizes benefits as the central management node. The selected central management nodes form the clusters with the nearest N nodes through the exchange of control packets.

However, the shape of the clusters and the content placement (refer to Section 3.1) within each cluster formed by these schemes differ from the design concept of the cluster in the proposed scheme. Therefore, applying these approaches directly to the proposed scheme may not achieve sufficient performance.

In this chapter, to ensure the feasibility of the proposed scheme to practical topologies, I propose a clustering algorithm in a practical topology. Moreover, I investigate the effectiveness of the proposed scheme and its dependence on the shape of topology/cluster through simulation evaluations using actual topologies.

5.2 Proposed scheme

In the aforementioned Section 3.2, the proposed scheme uniformly distributes content with equal distance between the same content by separating clusters with the same size and the same content placement. As a result, fair content retrieval latency and high cache utilization are achieved regardless of the consumer's location, because the cached contents in the consumer's exploration range, i.e., cluster size, do not duplicate regardless of the consumer's location. In other words, in the design of the proposed clustering approach, content is distributed without redundancy into a constant range around each CR, so that each CR logically behaves as the center of the cluster such as a circular shape. Hence, the problem is to determine a hash value assignment that achieves this content placement on the practical topology, and I propose the clustering algorithm to address this problem.

I consider a certain topology of ICN, denoted by the graph $G = (V, E)$. Let

V denote the set of CRs and E the set of communication links connecting them. Each CR is assigned one CRID k (i.e., hash value) from a set of CRIDs K ($K = 1, 2, ..., K$). Note that the number of CRIDs represents the cluster size. The assignment of CRIDs to each CR is represented as $V = v_1^k, v_2^k, ..., v_V^k$. The proposed clustering algorithm aims to find the assignment that minimizes a certain $Cost$. The $Cost$ is defined by the total sum of $Dist$ for each CR and is expressed in Equation 5.1.

$$Cost = \sum_{v \in V} Dist_v \tag{5.1}$$

Here, $Dist$ is given by Equation 5.2.

$$Dist_{v^k} = \sum_{n \in K, n \neq k} D_n \tag{5.2}$$

where D is the distance from CR v assigned CRID k to the closest CR assigned CRID n ($n \neq k, n \in K$), and $Dist$ for CR v is defined by the sum of D. Consequently, minimizing $Cost$ leads to minimizing the distance between CRs assigned with different CRIDs, thus forming a circular cluster centered on each CR.

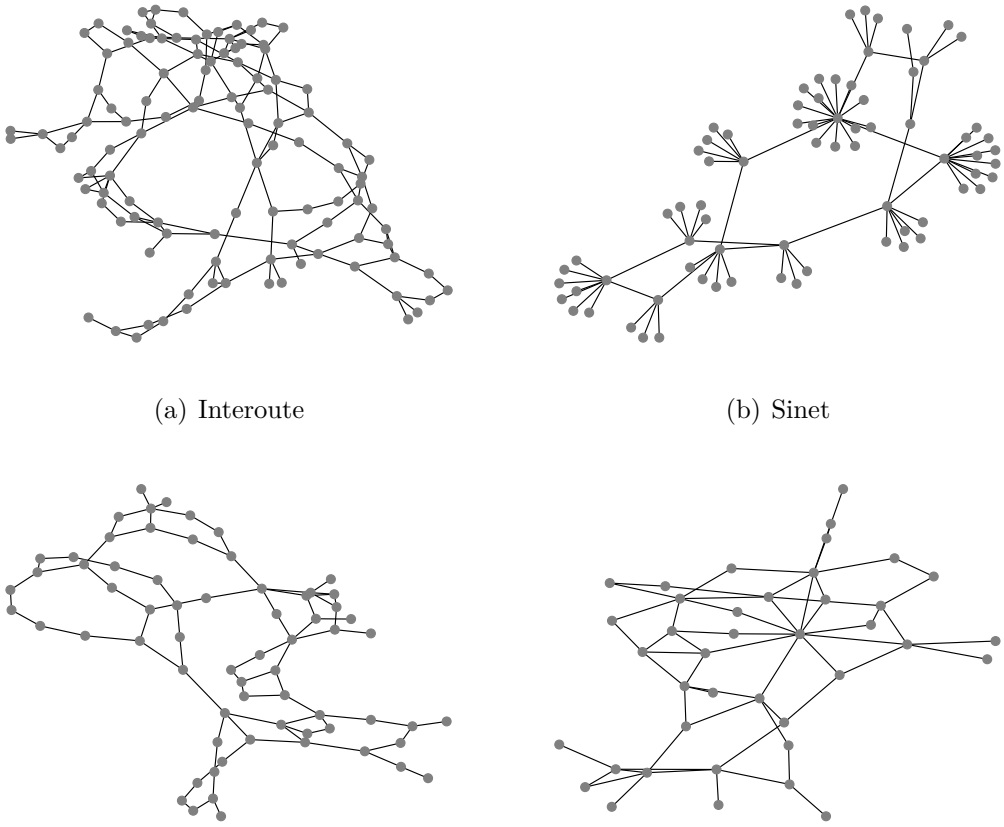
There are various ways to minimize this cost. In the exhaustive search approach, it is necessary to consider from V^K possible combinations (to assign one CRID chosen from the set of CRID K to each CR). Since this approach incurs impractical computational costs, I recommend using techniques such as simulated annealing or genetic algorithms to solve this minimization problem. However, even then, it may still take some computation time, so it is impractical to run the proposed algorithm in real time. To address this issue, network administrators should assume and preprocess clustering patterns in advance. This proposed algorithm, which does not strictly divide the network, i.e., the clusters are not uniform in shape and overlap, is applicable to any network regardless of any cluster size. Moreover, the distance D between CRs used in Equation 5.2 can be applied to various clustering scenarios by adopting similarity metrics proposed in other studies.

Table 5.1: Simulation parameters

Link bandwidth	100 [Mbps]
Propagation delay time	5 [ms]
Amount of Contents	128
Content size	64 [chunk]
Chunk size	1000 [Byte]
CS size on CR	128 [chunk]
Zipf α	0.6, 1.0, 1.4, 1.8

5.3 Simulation model

I evaluate the proposed and conventional schemes comparatively in a practical network topology where content popularity changes through simulations using Network Simulator ns-3 ver. 3.40 [40] with the implementation of the proposed scheme. I used the actual topologies of various shapes such as the Interoute topology of 110 nodes, the Sinet topology of 74 nodes, the Missouri topology of 67 nodes, and the Geant topology of 40 nodes, from the Internet Topology Zoo [42], as shown in Fig. 5.1. Interoute consists of a large number of nodes, which is similar to the grid topology used in Chapters 3 and 4. In contrast, Geant consists of fewer nodes. Additionally, Sinet and Missouri were chosen for their medium-sized nodes and distinct characteristics. The former mainly connects nodes in a star configuration, which shortens the average distance between reachable nodes. The latter mainly connects nodes in a ladder configuration, which lengthens the average distance between reachable routers. Since Fig. 5.1 shows the relationship of pop-level routers, I defined each node as CR and placed producers and consumers on each CR. The content was randomly placed on each producer. The parameters used in the simulation are summarized in Table 5.1. Each consumer sent Interest packets requesting content toward the producer at normal distribution intervals with an average value of 1.0 seconds. The requested content was determined on the basis of the content popularity, in which P2P content was generally known to follow a Zipf-mandelbrot distribution [41], and I gave q a fixed value of 5 and changed the content popularity with α to avoid the



(a) Interoute (b) Sinet

(c) Missouri (d) Geant

Figure 5.1: Simulation topologies

complexity of the discussion. Furthermore, I assumed no packet loss occurs so we can focus on the fundamental characteristics of the dynamic clustering approach. The simulation was performed for 270 seconds. I set the Zipf parameter α to 1.0 at the start of the simulation as shown in Fig. 5.2. α changed to 0.6 at 30 seconds after the simulation started, in which a wider range of content is requested, to 1.8 at 90 seconds, to concentrate on the requested content, and after that, it decreases by 0.4 every 60 seconds back to 1.0.

In this simulation, I compared and evaluated the effectiveness of five representative schemes: LCE, HR [39], HRC [13], Static [18], and proposed (Dynamic). Note that the HRC scheme uses the k-split algorithm with the number of hops

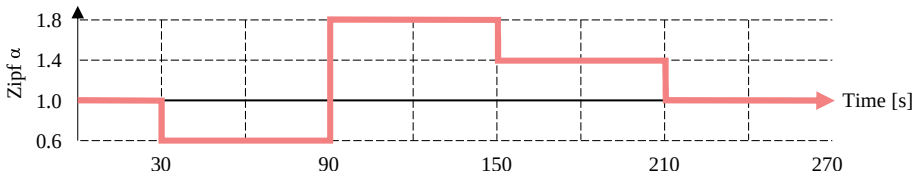


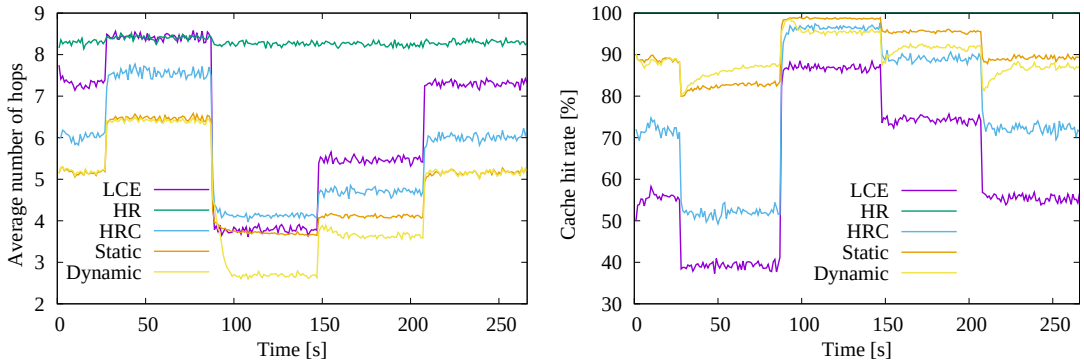
Figure 5.2: Fluctuation of zipf α

Table 5.2: Simulation parameters for each scheme

Scheme	Parameter	Interoute	Sinet	Missouri	Geant
HRC	Cluster size	5	4	4	3
Static		16			
Dynamic	Initial cluster size	16			
	Migratable cluster size	4, 9, 16, 36			
	Lower/upper threshold	70/100	70/80	50/60	40/50

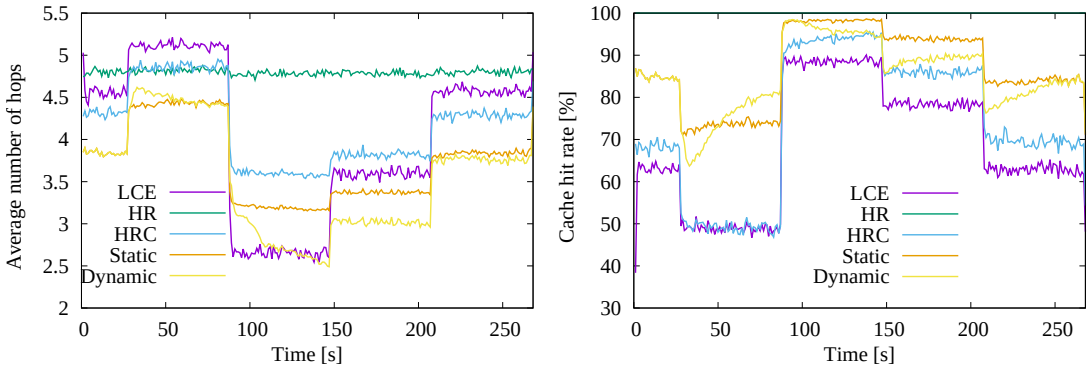
as similarity metrics for clustering and forms k clusters. The Dynamic and Static scheme used the clustering algorithm described in Sec. 5.2 and set the reclustering interval to 3 s and floodingTTL to 6 hops. The other parameters for each scheme used in the simulation are summarized in Table 5.2. These settings were the appropriate value for an α of 1.0 at the start of the simulation.

Furthermore, the average number of hops needed to retrieve content, cache hit rate, and advertisement rate were used as evaluation indices to discuss the effectiveness of the proposed scheme. The average number of hops focused on content retrieval time, which was defined as the total number of hops during the time when all consumers retrieved content divided by the total number of requests for all consumers. The cache hit rate focused on cache efficiency, which was defined as the total number of cache hits on all CRs divided by the total number of requests for all consumers. The advertisement rate focused on communication overhead, which was defined as the amount of advertisement packets divided by the total amount of traffic. In this study, I assumed the average name length is 30 bytes, and the size of the advertisement packet which includes the content name, the flooding limit, and the flag bit that indicates the cache information (newly cached/discarded), is the same as the Interest packet.



(a) Average number of hops (b) Cache hit rate

Figure 5.3: Effect of Interoute

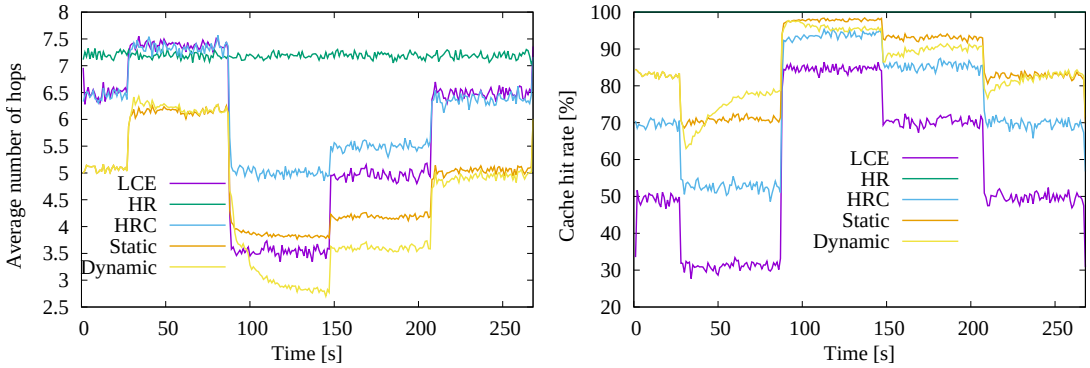


(a) Average number of hops (b) Cache hit rate

Figure 5.4: Effect of Sinet

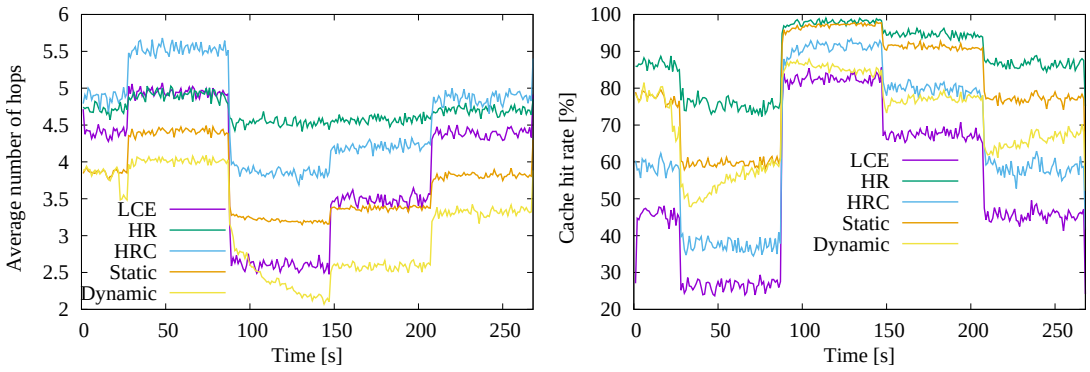
5.4 Simulation results

In this section, I evaluate the proposed scheme on actual topologies of various shapes to discuss its feasibility and effectiveness. Figures 5.3, 5.5, 5.4, and 5.6 commonly show the average number of hops and cache hit rate as a function of time on each topology. These indicate that the trend is almost the same as the results for the grid topology shown in Section 4.4.1, and the Dynamic scheme always maintains the high cache hit rates and reduces the average number of hops regardless of the shape of topologies. In addition, regarding communication overheads, the Dynamic scheme achieves smaller advertisement rates than the



(a) Average number of hops (b) Cache hit rate

Figure 5.5: Effect of Missouri



(a) Average number of hops (b) Cache hit rate

Figure 5.6: Effect of Geant

Static scheme (Table 5.3). Therefore, the proposed scheme is effective even in practical network topologies.

5.5 Conclusion

The design of the proposed schemes and their effectiveness have been discussed on the basis of a simple topology in Chapters 3 and 4. However, to apply the proposed scheme to practical networks, it is essential to establish a clustering scheme for practical networks and evaluate its performance regarding topology

Table 5.3: Advertisement rate on each topology

	Interoute	Sinet	Missouri	Geant
Static	6.59 %	7.80 %	5.82 %	5.38 %
Dynamic	4.76 %	5.20 %	3.88 %	3.52 %

dependence. Therefore, I proposed a clustering scheme for practical networks and evaluated the feasibility and effectiveness of the proposed scheme through simulations conducted on various actual topologies. Simulation evaluations have indicated that the proposed scheme can reduce the delivery latency while consistently maintaining a high cache hit rate in practical networks.

6 Conclusion and future work

In this chapter, I conclude this dissertation and address future work.

6.1 Conclusion

The primary usage of today's network is for delivering video content, such as Video on Demand (VoD), which constitutes approximately 80% of the total network traffic. It is anticipated that in the near future, the proliferation of IoT (Internet of Things) services will lead to the influx of a large volume of IoT content into the network. Therefore, solutions are needed to support the current network usage, which revolves around content distribution. A new network architecture called Information-Centric Networking (ICN) has attracted attention as a potential solution. ICN is designed with a focus on the concept that, when consumers access content, they are indifferent to the identity of the content provider. This design allows for directly exploring content in the network without being dependent on the content provider by changing the destination of a content request from the traditional Internet protocol (IP) address to the content name. To take advantage of this design concept, content routers (CRs), which are intermediate routers, are assigned the roles of both forwarding and caching content. This allows replicated content to be distributed into the network while acting as substitutes for content providers. Therefore, ICN can handle the content requests by using CRs without the intervention of content providers, which contributes to balancing server and network loads and reducing content delivery latency.

However, the cache size of CR is significantly smaller than the huge amount of content worldwide, so it is naturally impossible to cache all content. Therefore, an important issue to be addressed by ICN is to consider how to use these cashes, since they significantly affect the performance of content retrieval. In recent years,

to address this issue, clustering schemes have been proposed. These schemes group several CRs (i.e., a cluster) in the network and apply efficient content placement and routing within each cluster. This allows for resolving the majority of consumer content requests by utilizing a near cluster. However, I believe that we can achieve higher efficiency by focusing on two points, which are still open for discussion:

1. How to efficiently use surrounding clusters. The routing design of these clustering schemes has not allowed forwarding outside the cluster for request resolution. Specifically, the cache exploration range is restricted to the cluster boundaries, and the requests for non-cached content within the cluster are forwarded directly to the producer. However, although a cluster has a large cache size, it cannot cache the huge amount content worldwide, so the caches in neighboring clusters should also be made available. Therefore, routing that is not restricted by cluster boundaries is necessary.

2. How to adapt to changes in content demand trends. The design of these clustering schemes aims to improve cache utilization and reduce delivery latency by retaining a sufficient amount of the main popular content in the caches of a cluster. However, in a practical environment, the content demand trends, i.e., the amount of the main popular content, will change over time. Therefore, the cluster size needs to be determined depending on the situation.

In this dissertation, as one solution for efficiently using caches in ICN to meet the recent demand for content distribution, I proposed a clustering scheme that considers issues 1 and 2. This proposed scheme efficiently delivers contents by making clusters that are efficiently clustered and distributed content in accordance with the network situation and by flexibly exploring valuable content from the clusters. Furthermore, I demonstrated the effectiveness of the proposed scheme through simulation evaluations.

Chapter 2 provided an overview of the inception of ICN, its fundamental operations, related works, and the two specific issues addressed in this dissertation.

Chapter 3 focused on issue 1 and proposed an efficient content distribution and exploration scheme for clustering schemes. This scheme uniformly distributes content to each cluster in a distributed caching manner and dynamically updates the routing table on the basis of collaboration among CRs across the cluster bound-

ary. As a result, consumers can explore the nearest content from surrounding clusters. In simulation evaluations, the proposed scheme demonstrated higher cache efficiency and lower content delivery latency than conventional schemes by using surrounding clusters/caches.

Chapter 4 focused on issue 2 and proposed a dynamic clustering scheme to adjust the cluster size in accordance with fluctuations in content demand trends. This scheme effectively estimates the appropriate cluster size by using a simple threshold-based algorithm on the basis of the frequency of cache updates in the cluster. As a result, it enables the construction of consistently appropriate clusters to adjust to shifting content demand trends. Simulation evaluations indicated that the proposed scheme reduces delivery latency while consistently maintaining a high cache efficiency in an environment with changing content demand trends.

Chapter 5 delved into the adaptability of the proposed scheme in practical networks. While the design and effectiveness of the proposed scheme were discussed on the basis of the simple topology in Chapters 3 and 4, applying the scheme to practical networks requires establishing a clustering scheme for their networks and evaluating its performance regarding topology dependence. Therefore, I proposed a clustering scheme for practical networks, and in simulation evaluations, the proposed scheme indicated sufficient applicability in practical networks.

In this dissertation, I solved the two specific issues 1 and 2 on clustering schemes in ICN. By deploying the proposed scheme on the network, clusters that automatically formed on the basis of content demand trends among consumers will contribute to improving quality of service regarding diverse content distribution in the future, as shown in Fig. 6.1. However, the following issues remain for future work.

6.2 Future work

As mentioned above, various caching schemes have been proposed to support the content distribution in ICN. The common direction of these designs is to take into account the content popularity for efficiently using the caches. However, as ICT technology further evolves, various services with content delivery formats that are beyond our imagination will appear in the future. As a result, these designs may

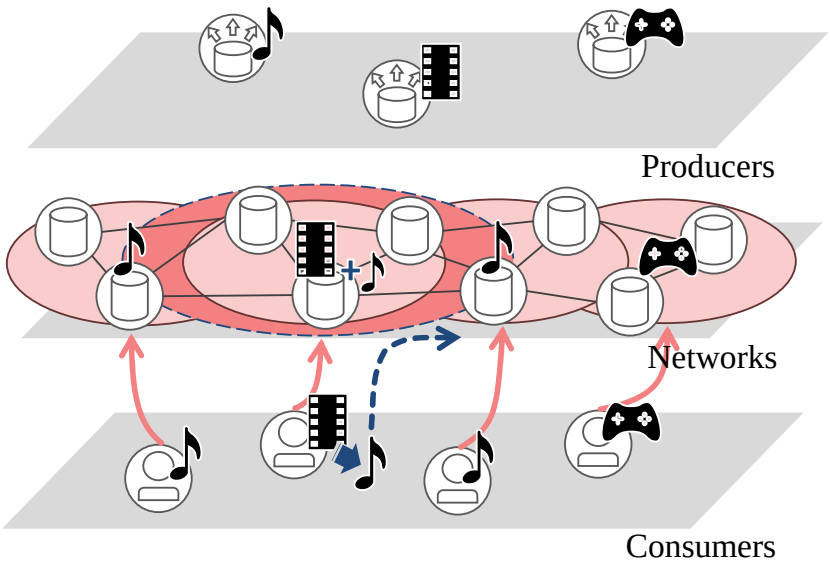


Figure 6.1: An overview of the proposed cluster-based ICN

not be sufficient to support the efficient content distribution of these services. For instance, IoT services, which have attracted attention in recent years, distribute content much smaller than video content, which is currently the main content distributed on the network. In automatic driving, which is a typical IoT service, to constantly determine the direction and speed of the car in accordance with the traffic conditions on the road, the server must collect and process the information around the car and feed control information to the car within an extremely short time. However, while almost all caching schemes prioritize caching the popular content for many consumers, they may not effectively cache the content that is very important for automated driving services and needs to be delivered as quickly as possible, such as traffic accident information. This is because these contents not only are used by far fewer consumers than publicly available video contents but also are generated suddenly and have limited time for utilization, so the importance of the contents is difficult to estimate on the basis of content popularity. From the above, it is important to consider how to utilize the caches in the future, considering such aspects on the service side as well as the consumer side.

Therefore, as a direction for future study, I will consider automatically forming

and layering clusters that can cache content in accordance with the type of service, on top of clusters for consumers. I expect that this approach will not only contribute to enhancing the quality of services provided on the network but will also support the introduction of services that have been difficult to realize in the past.

References

[1] L. Roberts, "The Arpanet and computer networks," *A history of personal workstations*, pp. 141–172, Jan. 1988. DOI:10.1145/61975.66916

[2] T. Berners-Lee and R. Cailliau, "WorldWideWeb: Proposal for a HyperText Project," [Online]. Available: https://www.w3.org/Proposal.html. Accessed on Dec. 12, 2023.

[3] Cisco, "Cisco Annual Internet Report (2018–2023)," [Online]. Available: https://www.cisco.com/c/ja_jp/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html. Accessed on Dec. 12, 2023.

[4] National Institute of Information and Communications Technology (NICT), "Beyond 5G/6G white paper version 1.0," [Online]. Available: https://beyond5g.nict.go.jp/images/download/NICT_B5G6G_WhitePaperEN_v1_0.pdf. Accessed on Jan. 12, 2023.

[5] A. Passarella, "A survey on content-centric technologies for the current Internet: CDN and P2P solutions," *Elsevier Computer Communications*, vol. 35, no. 1, pp. 1–32, Jan. 2012. DOI:10.1016/j.comcom.2011.10.005

[6] S. Androutsellis-Theotokis and D. Spinellis, "A survey of peer-to-peer content distribution technologies," *ACM Computing Surveys*, vol. 36, no. 4, pp. 335–371, Dec. 2004. DOI:10.1145/1041680.1041681

[7] W. Yu, F. Liang, X. He, W.G. Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the Internet of Things," *IEEE Access*, vol. 6, pp. 6900–6919, Nov. 2017. DOI:10.1109/ACCESS.2017.2778504

[8] V. Jacobson, D.K. Smetters, J.D. Thornton, M. Plass, N. Briggs, and R. Braynard, "Networking named content," *Communications of the ACM*, vol. 55, no. 1, pp. 117–124, Jan. 2012. DOI:10.1145/2063176.2063204

[9] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K.C. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, Jul. 2014. DOI:10.1145/2656877.2656887

[10] A. Ioannou and S. Weber, "A survey of caching policies and forwarding mechanisms in information-centric networking," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2847–2886, May 2016. DOI:10.1109/COMST.2016.2565541

[11] S. Arshad, M.A. Azam, M.H. Rehmani, and J. Loo, "Recent advances in information-centric networking-based internet of things (ICN-IoT)," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2128–2158, Apr. 2019. DOI:10.1109/JIOT.2018.2873343

[12] M. Meddeb, A. Dhraief, A. Belghith, T. Monteil, K. Drira, and S. Al-Ahmadi, "Named data networking: A promising architecture for the internet of things (IoT)," *International Journal on Semantic Web and Information Systems*, vol. 14, no. 2, pp. 86–112, Apr. 2018. DOI:10.4018/IJSWIS.2018040105

[13] V. Sourlas, I. Psaras, L. Saino, and G. Pavlou, "Efficient hash-routing and domain clustering techniques for information-centric networks," *Elsevier Computer Networks*, vol. 103, pp. 67–83, Jul. 2016. DOI:10.1016/j.comnet.2016.04.001

[14] C. Li and K. Okamura, "Cluster-based in-networking caching for content-centric networking," *International Journal of Computer Science and Network Security*, vol. 14, no. 11, pp. 1–9, Nov. 2014. http://paper.ijcsns.org/07_book/201411/20141101.pdf.

[15] B. Alahmri, S. Al-Ahmadi, and A. Belghith, "Efficient pooling and collaborative cache management for NDN/IoT networks," *IEEE Access*, vol. 9, pp. 43228–43240, Mar. 2021. DOI:10.1109/ACCESS.2021.3066133

[16] H. Yan, D. Gao, W. Su, C.H. Foh, H. Zhang, and A.V. Vasilakos, "Caching strategy based on hierarchical cluster for named data networking," *IEEE Access*, vol. 5, pp. 8433–8443, Mar. 2017. DOI:10.1109/ACCESS.2017.2694045

[17] S. Traverso, M. Ahmed, M. Garetto, P. Giaccone, E. Leonardi, and S. Niccolini, "Temporal locality in today's content caching: Why it matters and how to model it," *SIGCOMM Computer Communication Review*, vol. 43, no. 5, pp. 5–12, Oct. 2013. DOI:10.1145/2541468.2541470

[18] M. Yoshida, Y. Ito, Y. Sato, and H. Koga, "A cluster-based cache distribution scheme in content-centric-networking," *Proc. ACM Conference on Information-Centric Networking*, pp. 196–197, Sept. 2018. DOI:10.1145/3267955.3269012

[19] M. Yoshida, Y. Ito, Y. Sato, and H. Koga, "Popularity-aware dynamic-clustering scheme for distributed caching in ICN," *Proc. ACM Conference on Information-Centric Networking*, pp. 192–193, Sept. 2022. DOI:10.1145/3517212.3559482

[20] M. Yoshida, Y. Ito, Y. Sato, and H. Koga, "Performance evaluation of popularity-aware dynamic-clustering scheme for distributed caching in ICN," *Proc. APSIPA Annual Summit and Conference*, pp. 185–190, Nov. 2022. DOI:10.23919/APSIPAASC55919.2022.9979928

[21] M. Yoshida, Y. Ito, Y. Sato, and H. Koga, "PopDCN: Popularity-aware dynamic-clustering scheme for distributed caching in ICN," *to appear in IEICE Transactions on Communications*, 10 pages, May 2024. DOI:10.23919/transcom.2023EBP3152

[22] A. Dominguez, O. Novo, W. Wong, and T. Valladares, "Publish/subscribe communication mechanisms over PSIRP," *Proc. IEEE International Conference on Next Generation Web Services Practices*, pp. 268–273, Oct. 2011. DOI:10.1109/NWeSP.2011.6088189

[23] T. Koponen, M. Chawla, B.G. Chun, A. Ermolinskiy, K.H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, pp. 181–192, Aug. 2007. DOI:10.1145/1282427.1282402

[24] H. Che, Y. Tung, and Z. Wang, "Hierarchical Web caching systems: Modeling, design and experimental results," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 7, pp. 1305–1314, Sept. 2002. DOI:10.1109/JSAC.2002.801752

[25] S. Arianfar, P. Nikander, and J. Ott, "On content-centric router design and implications," *Proc. ACM SIGCOMM the Re-Architecting the Internet Workshop*, pp. 1–6, Nov. 2010. DOI:10.1145/1921233.1921240

[26] I. Psaras, W.K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," *Proc. ACM Workshop on Information-Centric Networking*, pp. 55–60, Aug. 2012. DOI:10.1145/2342488.2342501

[27] N. Laoutaris, H. Che, and I. Stavrakakis, "The LCD interconnection of LRU caches and its analysis," *Performance Evaluation*, vol. 63, no. 7, pp. 609–634, Jul. 2006. DOI:10.1016/j.peva.2005.05.003

[28] N. Laoutaris, S. Syntila, and I. Stavrakakis, "Meta algorithms for hierarchical web caches," *Proc. IEEE International Performance Computing and Communications Conference*, pp. 445–452, Apr. 2004. DOI:10.1109/PCCC.2004.1395054

[29] K. Cho, M. Lee, K. Park, T. T. Kwon, Y. Choi, and S. Pack, "Wave: Popularity-based and collaborative in-network caching for content-oriented networks," *Proc. IEEE Conference on Computer Communications Workshops*, pp. 316–321, May 2012. DOI:10.1109/INFCOMW.2012.6193512

[30] E.J. Rosensweig and J. Kurose, "Breadcrumbs: Efficient, best-effort content location in cache networks," *Proc. IEEE Computer and Communications Societies*, pp. 2631–2635, Apr. 2009. DOI:10.1109/INFCOM.2009.5062201

[31] Y. Li, T. Lin, H. Tang, and P. Sun, "A chunk caching location and searching scheme in content centric networking," *Proc. IEEE International Conference on Communications*, pp. 2655–2659, Jun. 2012. DOI:10.1109/ICC.2012.6363958

[32] L. Gong, "Intelligent forwarding strategy based on online machine learning in named data networking," *Proc. IEEE Trustcom/BigDataSE/ISPA*, pp. 1288–1294, Aug. 2016. DOI:10.1109/TrustCom.2016.0206

[33] L. Wang, S. Bayhan, J. Ott, J. Kangasharju, A. Sathiaseelan, and J. Crowcroft, "Pro-diluvian: Understanding scoped-flooding for content discovery in information-centric networking," *Proc. ACM Conference on Information-Centric Networking*, pp. 9–18, Sept. 2015. DOI:10.1145/2810156.2810162

[34] T. Mick, R. Tourani, and S. Misra, "MuNCC: Multi-hop neighborhood collaborative caching in information centric networks," *Proc. ACM Conference on Information-Centric Networking*, pp. 93–101, Sept. 2016. DOI:10.1145/2984356.2984375

[35] M. Lee, J. Song, K. Cho, S. Pack, T. Kwon, J. Kangasharju, and Y. Choi, "Content discovery for information-centric networking," *Elsevier Computer Networks*, vol. 83, pp. 1–14, Jun. 2015. DOI:10.1016/j.comnet.2014.10.006

[36] W. Wong, L. Wang, and J. Kangasharju, "Neighborhood search and admission control in cooperative caching networks," *Proc. IEEE Global Communications Conference*, pp. 2852–2858, Dec. 2012. DOI:10.1109/GLOCOM.2012.6503549

[37] H.M. Ju and L. Hyesook, "Cache sharing using Bloom filters in named data networking," *Journal of Network and Computer Applications*, vol. 90, pp. 74–82, Jul. 2017. DOI:10.1016/j.jnca.2017.04.011

[38] S. Nayak, R. Patgiri, and A. Borah, "A survey on the roles of Bloom Filter in implementation of the Named Data Networking," *El-*

sevier Computer Networks, vol. 196, art. no. 108232, Sept. 2021. DOI:10.1016/j.comnet.2021.108232

[39] L. Saino, I. Psaras, and G. Pavlou, "Hash-routing schemes for information centric networking," *Proc. ACM SIGCOMM workshop on Information-centric networking*, pp. 27–32, Aug. 2013. DOI:10.1145/2491224.2491232

[40] G.F. Riley and T.R. Henderson, "The ns-3 Network Simulator," *in Modeling and Tools for Network Simulation*, ed. K. Wehrle, M. Güneş, J. Gross, pp. 15–34, Springer, Berlin, Heidelberg, 2010. DOI:10.1007/978-3-642-12331-3_2

[41] M. Hefeeda and O. Saleh, "Traffic modeling and proportional partial caching for peer-to-peer systems," *IEEE/ACM Transactions on Networking*, vol. 16, no. 6, pp. 1447–1460, Mar. 2008. DOI:10.1109/TNET.2008.918081

[42] S. Knight, H. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, Oct. 2011. DOI:10.1109/JSAC.2011.111002