Cluster-based In-networking Caching for Content-Centric Networking

Chengming LI[†] and Koji OKAMURA[††],

[†]Department of Advanced IT, Graduate School of ISEE, Kyushu University
[††]Research Institute for Information Technology, Kyushu University.
Hakozaki 6-10-1, Higashi-ku, Fukuoka-shi, 812-8581, JAPAN

Summary
With the Internet architecture changing from host-centric communication model to content-centric model, Content Centric Networking (CCN) has emerged. One distinctive feature of CCN infrastructure is in-networking caching. As cache capacities of routers are relatively small compared with delivered data size, one challenge of in-networking caching is how to efficiently use the cache resources. In this paper, we proposed a cluster-based in-networking caching mechanism to improve the cache hit ratio and reduce caching redundancy for CCN. We designed the improved K-medoids cluster algorithm to cluster the whole network into *k* clusters and Virtual Distributed Hash Table (VDHT) to efficiently control and manage the resources stored in each cluster. We also proposed different policies for intra cluster routing and inter cluster routing to effectively forwarding requests. Compared with representative on-path caching schemes and hash scheme by simulation, we concluded that our cluster-based in-networking caching mechanism can improve the cache hit ratio and reduce link load of networks.

Key words:
Content-Centric Networking; In-networking Caching; Cluster; Cache Diversity;

1. Introduction

With the Internet architecture changing from host-centric communication model to content-centric model, several Information-Centric Network (ICN) architectures have been proposed, e.g., TRIAD [1], DONA [2], CCN/NDN [3]. As the CCN/NDN (Content-Centric Networking/ Named Data Networking) architecture is the most popular one, in this paper, we carry our work under the framework and the terminology of CCN/NDN. Furthermore, we use CCN instead of NDN for the method we designed is applicable to a wider class of CCN designs.

In CCN, contents are retrieved directly by their names, instead of locations. CCN is designed inherently focused on content distribution rather than host-to-host connectivity. Content in CCN is distributed in a scalable, cost efficient and secure manner. The change from host-centric to content-centric has several attractive advantages, such as network load reduction, low dissemination latency, and energy efficiency [3]. While the general infrastructure of CCN is in-networking caching, which allows any elements in the network to store content temporarily acting as servers.

The researches about catching have been widely carried out in the past [4]. While catching in CCN has its own content-oriented features. First of all, caching is a native property of routers in CCN. In CCN, request catching and content catching should be handled at a same network layer. That makes content retrieval and replacement be considered at line speed [5]. More clearly, a router should check if its local content store has the requested content before it sends a request to next hop. Secondly, the placements of the ubiquitous caches are arbitrary, but not hierarchical, which make caching in CCN different from Web-catching and Content Delivery Network (CDN). At last, as content chunks in CCN are identified by the unique names, different applications could use same cache space in a router at the same time. This is the most basic feature of catching in CCN, which make it different from web, CDN and P2P.

In-networking catching is the distinctive feature of CCN infrastructure and plays an important role in terms of system performance. In-network caching mechanism can avoid wasting network bandwidth due to the repeated delivery of popular content. Additionally, it can reduce response time for content by placing the content closer to users. The challenges surrounding in-networking caching involves cache placement, cache replacement and network cache model, etc. However, Kutscher, et al [6] define three key issues which influence the performance of in-networking caching system, i.e., cache placement, content-placement, request-cache routing. Cache placement mainly focus on deciding which nodes are supposed to upgrade for in-networking caching in a domain, which are mainly related with the whole network planning, such as, the network topology, traffic and positions [7], [8]. As for content-placement, it is an issue about the distribution policy of contents across in-networking caches in a domain. However, request-cache routing solves the problem of

Manuscript received November 5, 2014
Manuscript revised November 20, 2014

actions took for a content request corresponding to node caches. Above all, in this paper, we focus on the content placement issue and the request-cache routing issue.

As in-networking caching is so important for CCN, lots of in-networking caching strategies have been presented till now. There are mainly three caching strategies for in-networking caching. The “on-path catching” strategy is the one which allows contents to be cached temporarily at nodes on the path from content providers to consumers. This strategy reduces bandwidth consumption and content retrieval time by allowing contents closer to consumers. However, it has been demonstrated that this strategy is not optimal as it may imply a high content replication that limits the maximum number of contents that can be cached inside a domain [9], [10]. Therefore, “off-path caching” is an alternative strategy that can avoid duplications and can significantly increase the overall hit radio [10]. While, the “off-path caching” limited the scalability of CCN for its per-content state required for routing. Hence, mixed techniques were proposed, like SCAN [11], which mix features of on-path and off-path techniques.

The idea in this paper is motivated by following considerations. Due to content popularity, often, the same content is accessed by many users, which makes network traffic exhibit high redundancy. Furthermore, even CCN enables individual nodes to reduce redundancy by managing a local cache, redundancy can freely appear across different nodes as the default ubiquitous LRU caching scheme and the support of multi-path routing. These two aspects motive us to consider that controlling the redundancy level is a critical issue to improving the systematic caching performance of CCN.

The goals, our scheme aim to achieve, are improving cache hit, which means reduction in bandwidth usage, reducing caching redundancy, efficient utilization of available cache resources, and balancing distribution of content among the available caches.

To achieve the above goals, we proposed the cluster-based in-networking caching for CCN. Through dividing nodes of network into clusters, we make sure that no cache redundancy happens in a cluster to improve cache diversity. In order to efficiently control and manage the state of cache in a cluster, we use a distributed hash table.

The main contributions of this paper are:

- We proposed cluster-based in-networking caching for CCN to improve cache hit, reduce cache redundancy and improve efficient utilization of available cache resources, and balance distribution of content among the available caches.
- We also designed two kinds of routing policies for cluster-based in-networking caching, i.e., inter-cluster routing and intra-cluster routing.
- We evaluate the effectiveness of cluster-based in-networking caching scheme by extensive simulations. The results show that our scheme balance the cache hit radio and link load compared with other schemes.

The rest of the paper is organized as follows. Section 2 presents a survey of related work and back ground. System model and assumptions are detailed in Section 3 and Section 4 introduced our proposal, cluster-based in-networking caching for CCN. Section 5 is the simulation evaluation. Finally, conclusions are summarized in Section 6.

2. Background

In this section, we will firstly introduce the architecture of Content-Centric Networking briefly. Then, we are going to focus on proposed schemes of inter-networking caching in CCN. Though these descriptions, we built a basic conception for our work.

2.1 Content-Centric Networking

In CCN, a node has three components: the Content Store (CS), the Pending Interest Table (PIT) and the Forwarding Information Base (FIB) [12]. And there are two types of packets [3] in CCN: Interest and Data. Consumers use Interests, which containing hierarchically structured content names, to require desired data. Data stored in nodes or servers are also named in hierarchical structures, e.g., a movie produced by Youtube may have the name /Youtube/movies/Example.rmvb, which similar to a URI.

Actions are taken as follows when an Interest packet comes in a node from some faces:

1) By longest prefix matching, check whether the required data stored in the CS or not, if exits, the router return corresponding Data packet to faces the Interest coming from;

2) If not exits, the node checks whether the PIT has the name of requested Data packet. If exits, it adds the faces that the Interest comes from into the corresponding entry of PIT;

3) If not exits, the node creates a new entry and adds it into the PIT. Then, the node forwards the Interest to face or faces according the FIB to retrieve the requested data.

Procedures will be taken as following when a Data packet received by a node:

1) Sends the Data packet to the face or faces that marked in the corresponding PIT entry, and delete the entry;

2) If the Data packet satisfies caching polices, the node store it into CS.

2.2 In-networking Caching in CCN

Caching has been studied for Web system, P2P systems, network system and so on, in order to improve performance of systems by reducing bandwidth usage, server load, and response time. Generally speaking, caching schemes can be classified into two types, i.e., centralized caching and decentralized caching. Centralized caching is one whose data are only distributed by a central node and request will be responded by this central node. One typical example is Web-caching. Centralized caching mechanisms do better in controlling and managing network resources. However, it not only increases communication overhead for updating the content location, but also reduces flexibility in terms of available cache locations. Decentralized caching mechanisms cache the content at any place of network and manage the cached data by servers or routers locally, e.g., in-networking caching in CCN. In-networking caching is one of main infrastructures of CCN. As it needs no communication overhead and its operation is location-independent, in-networking caching improves system performance. In CCN, in-networking caching schemes are mainly divided into three categories: on-path caching, off-path caching [13], and hybrid techniques.

On-path caching schemes have already been studied in the past [15], which focus on the issue of cache placement [7, 16]. In CCN, Data packets in on-path caching schemes are stored in any on-path nodes [3] or a subset of traversed nodes [9, 14] as they travelling through the network. Interests are delivered according to the defined forwarding policies and Data packets will be returned in the inverse way of Interests. Representative on-path routing schemes are Leave Copy Everywhere (LEC) [12], Leave Copy Down (LCD) [15], ProbCache [14], Centrality-based caching [9]. The problems of the popularity-driven content caching [17], content location [18] etc. are attracted lots of attentions. Even the scalability of on-path caching is strong, it limits cache hits due to redundant caching of contents.

Data packets in network using off-path caching are cached to node according the defined rules. Interests must be forwarded in the same rules, as Data packets are traveling the reverse ways of Interests. Generally, Interests are handled by nodes in network cooperatively [19][20]. Author Rosensweig et al. [21] proposed a method named “Breadcrumbs”. By additionally storing minimal information regarding caching history, they developed a content caching, location, and routing system that adopts an implicit, transparent, and best-effort approach towards caching. Hash technologies are also studied for in-networking caching of CCN. Author Saino et al. [13] designed five different hash-routing schemes which exploited in-network caches without requiring network routers to maintain per-content state information. A domain is considered as a whole by these five schemes. In contrast to on-path caching, off-path caching owns a higher cache hits, but has limited scalability due to per-content state required for routing.

Hybrid techniques of on-path caching and off-path caching are also explored. By exploiting nearby and multiple content copies for the efficient delivery, SCAN [11] exchanges the information of the cached contents using Bloom filter. Compared with IP routing, SCAN can offer reduced delivery latency, reduced traffic volume, and load balancing among links. Hybrid techniques are supposed to mix features of on-path and off-path techniques and balance the performance of scalability and cache hits.

3. Problem Analysis

As mentioned above, both on-path caching and off-path caching have advantages and disadvantages. Redundancy of contents caching makes on-path caching have limited cache hits even it do well in scalability. Oppositely, the shortness of off-path caching is limited scalability which is caused by per-content state required for routing. Problems of on-path caching and hash caching are depicted in Fig. 1 and Fig. 2 respectively.

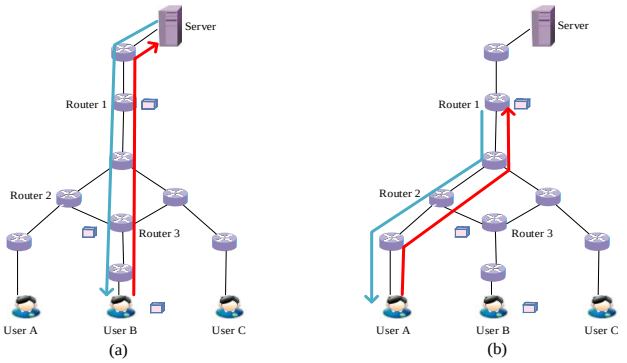

Fig. 1 Problem analysis of on-path caching.

Fig. 1 shows the problem happens in on-path caching. The red line represents the request route and the blue line indicates the data response route. In Fig. 1(a), User B sends an Interest packet to request Data in the red line.

After receiving Interest, Server returns Data packet travelling the reverse path of Interest to B. During that period, we assume that copies of that Data will be cached in Router 1 and Router 3 according the on-path caching methods. User B also stores the Data packet. When another user, User A, wants to request the same Data packet as B does, problem happens, as shown in Fig. 1(b). When the Interest sent by User A comes in Router 2, it is delivered to the direction of Router 1, and the request is responded by Router 1, as previous stored the request Data. However, we can see that, Router 3 also store the request Data packet and it is closer to Router 2 compared with Router 1. Ideally, we hope the closer router, Router 3, to response the request, which will reduce the link load of network.

Hash caching schemes proposed by Saino et al. [13] can solve the above problem. However, it is more suitable for small scale network. When use them in large scale networks, there are also problems. Fig. 2 demonstrates one problem happens in hash caching schemes. User sends an Interest whose request Data is previously stored in Router 2 and in Server, of course. When the Interest comes in Router 1, according to hash schemes, it is delivered to Router 2 and is responded by Router 2, rather than the Server, which is closer to Router 1.

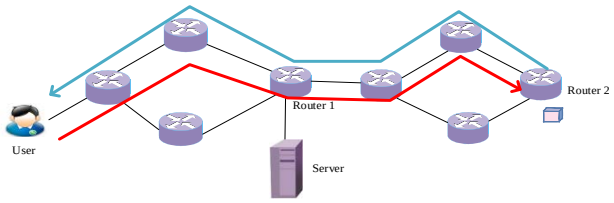

Fig. 2 Problem analysis of hash caching.

To solve above problems, the goals we are going to achieve are as follows:

- Improving cache hit: increase of cache hit means reduction in bandwidth usage;
- Reducing caching redundancy: caching has been traditionally used to reduce traffic redundancy; Efficient utilization of available cache resources can improve cache diversity and routing robustness;
- Balancing distribution of content among available caches.

4. Cluster-based In-networking Caching

As mentioned earlier, the same content will be accessed by many users due to popularity. Thus, network traffic exhibits high redundancy. Although CCN allows individual nodes manage local caches in order to reduce redundancy, redundancy can freely appear across different nodes, i.e., different nodes store copies of the same content. This is caused by the default ubiquitous LRU caching scheme and the support of multi-path routing. Motivated by these, we will introduce our solution, Cluster-based In-networking Caching, in this section.

4.1 Overview

We assume that topologies of networks are plane. Let $G=(V,E)$ be the graph representing the network where V ($|V|$ = N, N is the number of nodes in network) are the nodes and E are the edges in the graph. By using cluster algorithm, we divide the network into K clusters. Furthermore, we design corresponding routing polices for inter-cluster and intra-cluster.

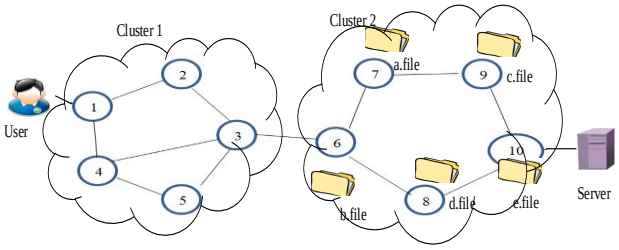

Fig. 3 An example of cluster-based in-networking caching

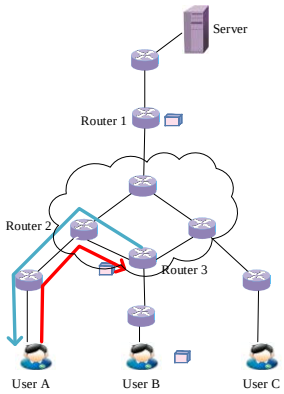

Fig. 4 Cluster-based caching solves the problem of on-path caching

In each cluster, there is no cache redundancy. To assure that, not only contents in a node are different from each other, but content in different nodes of a cluster are not same. However, content stored in different clusters could be same. Through this rule, we increase the cache diversity of the network. As shown in Fig. 3, there are two clusters in the network, Cluster 1 and Cluster 2. In Cluster 2, different nodes cache different content, while content cached by Cluster 1 can be same with content stored in Cluster 2.

Virtual distributed hash table is used to effectively control and manage chunks caching in each cluster. Nodes in a cluster use a same hash function to calculate location of caching node for a specific content.

Cluster-based in-networking caching can solve the problems described in Section 3. The problem, which happens in Fig. 1, can be solved as shown in Fig. 4. Here we assume the Router 2 and Router 3 are in the same cluster. After User B captured Data, same as Fig. 1(a), User A sends Interest to request the same Data. When the Interest delivered to Router 2, Router 2 forwards the Interest to Router 3 by the distributed hash routing table. Router 3 sends back the Data packet.

Fig. 5 details how cluster-based caching solves the problem happens in hash caching schemes. Firstly, the network is divided into two clusters, Cluster 1 and Cluster 2. Router 1, Router 3 and Router 4 are in Cluster 1, while Router 2 is in Cluster 2. When Router 4 receives the Interest sent by User, it firstly calculates the location of caching node for the requested Data packet, which is assumed Router 3. Then Router 2 forwards the Interest to Router 3. After receiving the Interest, Router 3 check whether the Data stored locally. If it not hits, Router 3 forwards the Interest to Server by the shortest path. At last, Server returns the Data packet by reverse path of the Interest.

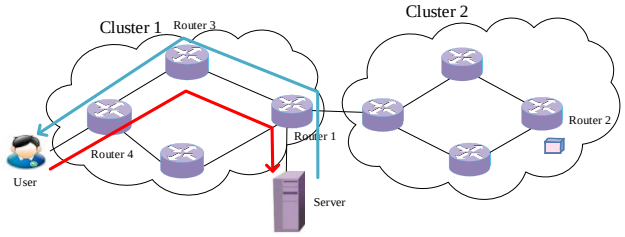

Fig. 5 Cluster-based caching solves the problem of hash caching

4.2 Improved K-medoids cluster algorithm

The cluster algorithm is descried in Algorithm 1, the improved K-medoids cluster algorithm. It needs two input parameters: K and G. The parameter K is currently assumed to be predefined based on the scale of the network, while the parameter $G=(V, E)$ demonstrates the graph of the network, $|V|=N$, N is number of network nodes and E is edges of the network. The result of the algorithm is K parts division of the network, i.e., K clusters. We replace the Euclidean distance in k-medoids clustering with new defined distance with Eq. 1 between two nodes. The reason is that Euclidean distance cannot reflect the real relationship between nodes in networks.

Eq. 1 is used to calculate distance value of any node i to centroid node of cluster k:

$$dis_k^i = \frac{d_i}{D} \times \frac{B}{b_i} \times \frac{C}{c_i} \times h_i \quad (1)$$

Where $i \neq k$ and D is average delay of all pair nodes in network. d_i is the delay from node i to node k by the shortest path; B is average bandwidth of all pair nodes in network, b_i is the average bandwidth from node i to node k by the shortest path; C is average cache size of all nodes on the path of each pair nodes in network; c_i is the cache size of nodes on the shortest path from node i to node k; h_i is the hops from node i to node k by the shortest path.

Algorithm 1 The improved K-medoids cluster algorithm

Input: K, predefined according the scale of networks; $G=(V,E)$, the graph of the network, $|V|=N$, N is number of nodes of the whole network.

Output: K clusters, divide the network into K parts

1 Delete nodes with only one edge;

2 Choose K nodes who have most number of edges; Set the K nodes as initial centroids of K clusters;

3 For each node i, calculate its dis_k^i from centroid node of cluster k ($k \in [1,K]$) by Eq. 1; add node i into the cluster with the smallest distance value;

4 For each cluster k, $\forall$ node $i, j \in$ k, calculate Dis_k^i by Eq. 2; set the node with smallest Dis_k^i value as the new centroids of cluster k;

5 Repeat 3 and 4 until centroids are not change.

The following equation is used to calculate average distance from node i to other nodes in cluster k.

$$Dis_j^i = \frac{1}{|k|} \sum_{i=1}^{i<|k|} \sum_{j=i+1}^{j\leq|k|} \frac{d_{ij}}{D} \times \frac{B}{b_{ij}} \times \frac{C}{c_{ij}} \times h_{ij} \quad (2)$$

4.3 Virtual Distributed Hash Table (VDHT)

To control and manage the contents stored in each cluster, each cluster holds only one Virtual Distributed Hash Table (VDHT). The VDHT is made up of CSs of all nodes in the cluster. Each CS of the node is indicated by the node ID. In a cluster, all nodes have a same hash function. When an Interest comes in a cluster through any nodes of the cluster, the receiving node calculates the location of caching node,

node ID, by the hash function according to the ID of requested chunk. Then the Interest will be delivered to the corresponding node.

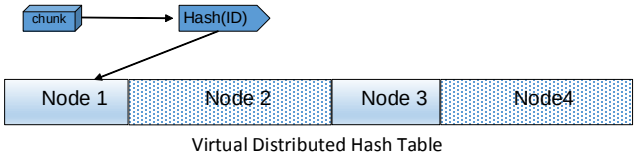

Fig. 6. An example of VDHT of a cluster

A simple VDHT of that cluster is shown in Fig. 6. To illustrate the components and functions of VDHTs, we assume there are four nodes in a cluster. The CSs of Node 1, Node 2, Node 3 and Node 4 construct VDHT as a whole. Each CS of the four nodes is indicated by node ID, Node 1, Node 2, Node 3 and Node 4, in the VDHT. When a Interest for chunk comes in the cluster through any of them, such as Node 3, then Node 3 calculates the node ID according the chunk ID by hash function, Node 1 for example. After that, the Interest will be forwarded from Node 3 to Node 1 and Node 1 checks the CS to make sure whether the requested chunk is cached here or not.

4.4 Intra-cluster and Inter-cluster forwarding

Before introducing routing polices, we first present the components of redesigned Interest packet and Data packet. An Interest packets consists content name, version, sequence number, cluster ID, node ID and other filed. Content name, version and sequence number is named as chunk ID. An example of Interest packet is shown in Fig. 7. Similarly, the header of a Data packet is similar with Interest.

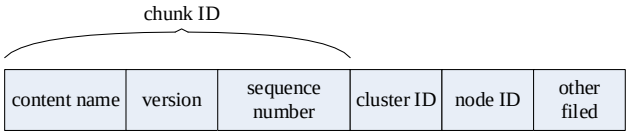

Fig. 7 An example of Interest packet

In cluster-based caching, we use Algorithm 2, the intra-cluster forwarding algorithm, to route Interest between nodes belonging to the same cluster.

When an Interest received by a node, the node firstly check if the cluster ID of Interest is same with its own cluster ID. If they are not same, it means that it is the first time that the Interest comes in the cluster. Then the node copies its cluster ID to *Interest.clusterID*, calculates a hash value using the Hash function according to the *Interest.chunkID* and gives it to *Interest.nodeID*, then forwards the Interest to the node with *Interest.nodeID*. However, if *Interest.clusterID* equals the cluster ID of the node, it means that this is intra-cluster forwarding. Then the node checks whether *Interest.nodeID* equals its own ID or not. This step is check whether the node is the location of the chunk that Interest request. If they are not equal, the node forwards the Interest to the node with *Interest.nodeID*. Conversely, the node is the location of the requested chunk, and then the node checks its CS to find the requested chunk. If the chunk exits, the node returns it, or the node forwards the Interest to the direction of the server.

Algorithm 2 The Intra-cluster Interest forwarding Algorithm

```
Input:  Interest;
Output: Forwarding decision
1  if Interest.clusterID ≠ my.clusterID then
2     Interest.clusterID ← my.clusterID;
3     Interest.nodeID ← Hash(Interest.chunkID);
4     Forward the Interest to the node with nodeID;
5  else
6     if Interest.nodeID ≠ my.nodeID then
7        Forward the Interest to the node with
         Interest.nodeID;
8     else
9        if the request Data cached in CS then
10          return Data;
11       else
12          Forward the Interest to Server;
13       End if
14    End if
15 End if
```

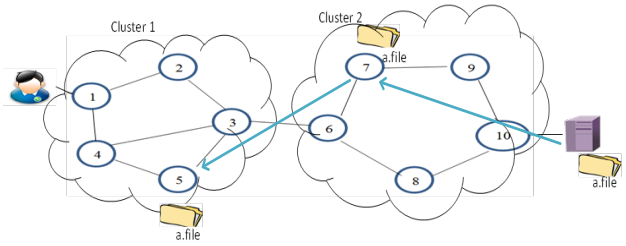

Fig. 8 An example of inter cluster routing

After Data packet generated, the Data packet is forwarded by reverse route of the Interest and stored in the node with *Data.nodeID* on the first cluster it passed through.

Among clusters, we use on-path caching. Here we use LCD. In inter-cluster forwarding case, when data already cached in current cluster, data packet will be cached in the next cluster data packet passed through. Fig. 8 shows an example of inter cluster routing. After someone has requested a.file, a.file was cached in node 7 in Cluster 2.

When other users request a.file through nodes in Cluster 1, a.file will be delivered to Cluster 1 and stored in node 5 as depicted.

5. Performance Evaluation

We evaluate the performance of cluster-based in-networking caching by extending the ccnSim [22], which is a CCN simulator based on OMNET++ [23]. For the comparison, we also implement Hash scheme, ProbCache, LCE on the ccnSim simulator.

5.1 Simulation Settings

In simulation, a network is modeled as a graph $G(n, p)$, where n is the number of nodes in the network and p is the probability of a connecting link exists between two nodes. GT-ITM [24] is used to generate a topology simulating the Internet, whose n = 50, p = 0.3. Links between nodes are characterized by their bandwidth and propagation delay. The bandwidth of each link is randomly chosen from a set $\{100\text{Mbs}, 150\text{Mbs}, 200\text{Mbs}\}$ and link propagation delays range from 1ms to 5ms. We divide the network into 6 clusters.

Table 1: Simulation parameters

Para	***Value***	Explanation
n	50	Number of nodes
p	0.3	Connectivity probability
b	{100,150, 200}Mbs	Link bandwidth
d	[1,5]ms	Link delay
α	0.9	content popularity distribution skewness
q	0.25	content popularity distribution skewness
Chunk size	10KB	CCN chunk size
Cache size	10GB	Cache size of each node
Catalog size	10^7 files	each file is 10^3 chunks
(Cache/ Catalog) ratio	$1*10^{-4}$	C/(|F|F)

We use the Mandelbrot-Zipf distribution model to calculate the content popularity. Unless otherwise specified, we set α=0.9 and q=0.25. There are two repositories which store the same content. Among the nodes, we randomly select 2 nodes which are connected to repository. The network has 10 client users which are connected to its border nodes. Users perform File-level requests according to a Poisson process with exponentially distributed arrival times at a 2 Hz rate.

All caching schemes have been evaluated assuming that the chunk size is 10KB; file size is about 10^3 chunks; catalog size is up to 10^7 files. We select cache sizes of 10 GB and keep the ratio of cache over catalog on the order of 10^{-4} ($Cache/Catalog = 10^{-4}$). The contents are evicted according to a Least Recently Used (LRU) policy (evicts the least recently used packet) and are cached by decision ALWAYS policy (caches every chunk it receives). For clarity, we list parameters described above and other parameters in Table 1.

In order to assess the performance of the cluster-based caching, we implement following schemes:

- Cluster: Cluster-based caching scheme proposed in this paper;
- Hash: Symmetric hash routing proposed in [13];
- ProCache: cache content along the path by probability;
- LCE: cache everything everywhere.

We compare the four schemes by focusing on two metrics: 1) cache hit radio, which represents the capability of the caching scheme to reduce the amount of redundant traffic; 2) link load, which is used to evaluate the efficiency of data transmission on network; 3) average data retrieve time, which denotes performance of network in user view.

Cache hit radio and link load are measured for four schemes under varying content popularity distribution skewness and cache over catalog radio. Content popularity distribution skewness is represented by parameter α and q of the Mandelbrot-Zipf.

5.2 Simulation Results

The simulation results are depicted from Fig. 9 to Fig. 12. Fig. 9 and Fig. 10 show the changing curves of cache hit ratio of four schemes with varying cache over catalog ratio and varying content popularity skewness. The affections of changing cache over catalog ratio and varying content popularity skewness on the link load of the network are represented on Fig. 11 and Fig. 12.

From the Fig. 9, we can get that with the increase of cache over catalog ratio, the cache hit radio of all schemes increased. Further, all the off-path schemes, Cluster-based caching and Hash routing, have higher cache hit ratio than on-path schemes, ProbCahe and LCE. More specifically,

Cluster-based caching scheme outperforms than ProCache and LCE, while it does worse than Hash routing scheme. This phenomenon is caused by that cluster-based caching using cluster method to improve the cache diversity of the network compared with ProCache and LCE, while hash routing has higher cache diversity than that of cluster-based caching by making all contents of the network stored are different.

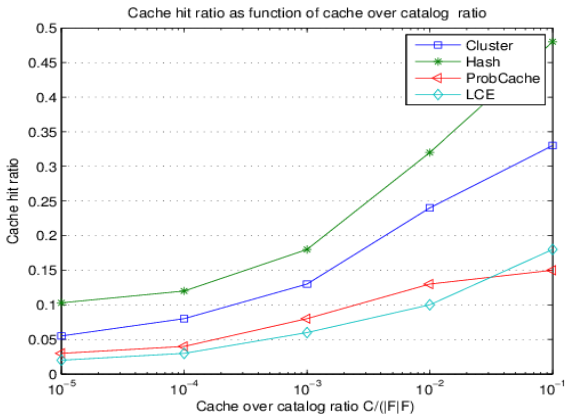

Fig. 9 cache hit ratio as function of cache over catalog ratio

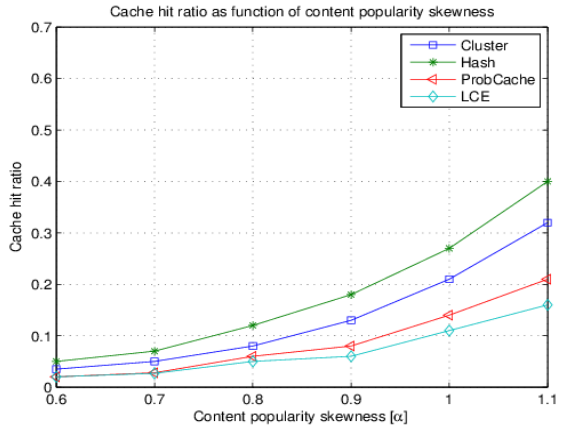

Fig. 10 Cache hit ratio as function of content popularity skewness

The results showed in Fig. 10 are similar with Fig. 9. However, the difference between them is that the gaps between lines in Fig. 10 are smaller than that in Fig. 9. From that we can conclude that cache over catalog ratio does more effect on the cache hit ratio than content popularity skewness does.

Fig. 11 shows that link load decreasing with increasing cache over catalog ratio. Conversely to cache hit ratio, all on-path schemes has lower values than off-path schemes have. In other words, on-path caching schemes outperform than off-path schemes in terms of link load. This is because the on-path schemes use the shortest path method at cost of throughput reducing to get data, which decrease link load.

Comparing the two off-path schemes, link loads of cluster-based caching are extraordinarily smaller than that of hash scheme. The reason for that is cluster reducing the redundant routes as Fig. 5 depicted.

Link loads are varying with changing content popularity skewness is shown in Fig. 12. As we can see the changing lines are similar with lines in Fig. 11.

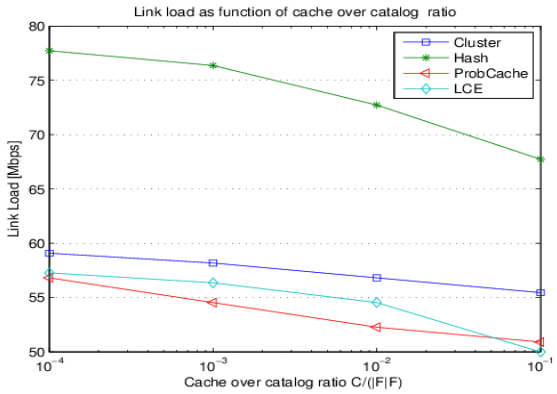

Fig. 11 Link load as function of cache over catalog ratio

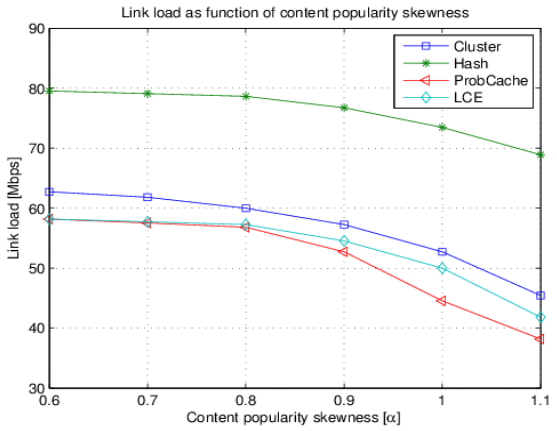

Fig. 12 Link load as function of content popularity skewness

6. Conclusion

In this paper, we proposed a cluster-based caching mechanism to improve cache hit, reduce caching redundancy, and balance distribution of content among available caches. In the cluster-based caching mechanism, we designed the improved K-medoids cluster algorithm to cluster the whole network into clusters. Virtual Distributed Hash Table (VDHT) was designed to efficiently control and manage the resources and contents stored in each cluster. We also proposed different policies for intra cluster routing and inter cluster routing to effective routing. Through simulations, our cluster-based in-networking

caching outperforms than on-path caching schemes, ProCache and LCE in terms of cache hit radio. It also do better in link load compared with hash schemes. To conclude, our cluster-based in-networking caching mechanism improves the cache hit ratio and reduce link load of the network.

We plan to extend our work on automatically generating the scale of clusters and the number of clusters of a network.

References

[1] Gritter M, Cheriton D R. An Architecture for Content Routing Support in the Internet[C]//USITS. 2001, 1: 4-4.

[2] Koponen T, Chawla M, Chun B G, et al. A data-oriented (and beyond) network architecture[J]. ACM SIGCOMM Computer Communication Review, 2007, 37(4): 181-192.

[3] Jacobson V, Smetters D K, Thornton J D, et al. Networking named content[C]//Proceedings of the 5th international conference on Emerging networking experiments and technologies. ACM, 2009: 1-12.

[4] Breslau L, Cao P, Fan L. Web caching and zipf-like distributions: evidence and implications. In: INFOCOM' 99. Proceedings of the IEEE 18th annual joint conference of the IEEE computer and communications societies, vol.1.IEEE; 1999.p.126–34.

[5] Rossi D, Rossini G, Caching performance of content centric networks under multi path routing (and more). Technical report, Telecom ParisTech; 2011.

[6] D. Kutscher and et al. Icn research challenges. IRTF, draft-kutscher-icnrg-challenges-00, Februady 2013.

[7] P. Krishnan, D. Raz, and Y. Shavitt. The cache location problem. IEEE/ACM Trans. Netw., 8(5), 2000.

[8] V. Pacifici and G. Dan. Content-peering dynamics of autonomous caches in a content-centric network. In IEEE INFOCOM, 2013.

[9] Chai W K, He D, Psaras I, et al. Cache “less for more” in information-centric networks [M]//NETWORKING 2012. Springer Berlin Heidelberg, 2012: 27-40.

[10] Saucez D, Kalla A, Barakat C, et al. Minimizing bandwidth on peering links with deflection in named data networking[J]. INRIA Sophia Antipolis Méditerranée, Tech. Rep, 2012.

[11] Lee M, Cho K, Park K, et al. Scan: Scalable content routing for content-aware networking[C]//Communications (ICC), 2011 IEEE International Conference on. IEEE, 2011: 1-5.

[12] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. Thornton, D. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos, et al. Named data networking (ndn) project. NDN-0001, Xerox Palo Alto Research Center-PARC, 2010.

[13] Saino L, Psaras I, Pavlou G. Hash-routing schemes for information centric networking[C]//Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking. ACM, 2013: 27-32.

[14] I. Psaras, W. K. Chai, and G. Pavlou. Probabilistic in-network caching for information-centric networks. In Proceedings ICN Sigcomm workshop, pages 55–60, New York, NY, USA, 2012. ACM.

[15] N. Laoutaris, H. Che, and I. Stavrakakis. The lcd interconnection of lru caches and its analysis. Perform. Eval., 63(7), July 2006

[16] J. Wang. A survey of web caching schemes for the internet. SIGCOMM Comput. Commun. Rev., 29(5), Oct. 1999.

[17] J. Li and et al. Popularity-driven coordinated caching in named data networking. In Proceedings of ANCS, New York, NY, USA, 2012. ACM.

[18] G. Tyson and et al. A trace-driven analysis of caching in content-centric networks. In Proc of ICCCN, 2012.

[19] R. Tewari, M. Dahlin, H. M. Vin, and J. S. Kay. Design considerations for distributed caching on the internet. In Proceedings of ICDCS, Washington, DC, USA, 1999.

[20] K. Katsaros, G. Xylomenos, and G. C. Polyzos. Multicache: An overlay architecture for information-centric networking. Comput. Netw., 55(4):936–947, Mar. 2011.

[21] Rosensweig E J, Kurose J. Breadcrumbs: Efficient, best-effort content location in cache networks[C]//INFOCOM 2009, IEEE. IEEE, 2009: 2631-2635.

[22] G. Rossini, D. Rossi, et al. Large scale simulation of ccn networks. Large scale simulation of CCN networks, pages 1–4, 2012.

[23] Omnet++ network simulation framework. http://www.omnetpp.org/, 2013.

[24] Zegura E W. GT-ITM: Georgia Tech internetwork topology models (software)[J]. Georgia Tech,” http://www. cc. gatech. edu/fac/Ellen. Zegura/gt-itm/gt-itm/tar. gz, 1996.

Chengming Li received the B.S. degree in software engineering from Dalian University of Technology and M.S. degree in computer application technology from Dalian University of Technology in 2009 and 2011, respectively. Now he is a Ph.D. candidate in Kyushu University, supported by China Governmental Scholarship. His research interests include virtualization technologies and future internet.

Koji Okamura is a Professor at Department of Advanced Information Technology and also at Computer Center, Kyushu University, Japan. He received B.S. and M.S. Degree in Computer Science and Communication Engineering and Ph.D. in Graduate School of Information Science and Electrical Engineering from Kyushu University, Japan in 1988, 1990 and 1998, respectively. He has been a researcher of MITSUBISHI Electronics Corporation, Japan for several years and has been a Research Associate at the Graduate School of Information Science, Nara Institute of Science and Technology, Japan and Computer Center, Kobe University, Japan. He is interested in Internet and Next Generation Internet, Multimedia Communication and Processing, Multicast/IPv6/QoS, Human Communications over Internet and Active Networks. He is a member of WIDE, ITRC, GENKAI, HIJK projects and Key person of Core University Program on Next Generation Internet between Japan and Korea sponsored by JSPS/KOSEF.