Consistent Hashing Based Cooperative Caching and Forwarding in Content Centric Network

Kyi Thar, Saeed Ullah, Choong Seon Hong
Department of Computer Engineering, Kyung Hee University, Republic of Korea
Email: {kyithar, saeed, cshong}@khu.ac.kr

Abstract—The original Content Centric Network (CCN) employs a simple caching scheme, Leave Copy Everywhere (LCE). However, this scheme is not efficient because cache redundancy reduces the storage capacity of the CCN network. In this paper, to resolve the issue of cache redundancy, we propose a cooperative caching decision and forwarding mechanism which is based on consistent hashing and virtual routers. We divide the Autonomous System (AS) into several groups of routers. The routers in the group cooperatively store the contents (Data) and also forward the requests (Interest) cooperatively in order to increase the caching performance of the CCN network. Finally, we evaluate our proposal by using a chunk-level simulator. The results show that the cache hit ratio of our proposed scheme is better than other proposed schemes.

Keywords—Cooperative Caching, Content Centric Network, Forwarding, Consistent Hashing, Virtual routers, Content popularity.

I. INTRODUCTION

Jacobson et al. have proposed the Content Centric Networks (CCN) or Named Data Networking (NDN) [1] which is a descendent of web caching [2]. The main feature of a CCN router is to store data temporarily and give a copy of the stored data to the users when they make a request. There are two types of packets in CCN, the Interest packet (request for the Data) and the Data packet (also known as content object) which is the reply packet for the requested Interest. Data is stored in Content Store (CS), in the form of a sequence of segments (called chunks).

The decision for storage in the cache is defined by the Cache Decision Strategy (CDS). The original caching scheme of CCN, Leave Copy Everywhere (LCE), is very simple. However, LCE is not efficient because it may store several copies of the same content on the request path. Since the storage space of each router is limited, this caching redundancy reduces the cache hit ratio. Therefore, caching redundancy must be reduced, particularly in neighboring routers. In the recent past some cache strategy mechanisms have been proposed in order to reduce the cache redundancy. The authors in [3] used Leave Copy Down (LCD) scheme to reduce the duplicate Data on the request path. Progressive Caching in [4] extended the LCD to cache popular chunks and solved the problem of unpopular items. In [5] and [6], the authors used probabilistic caching schemes in which the probability of caching the Data increases as the content travels nearer towards the user on the request path. In [7], the authors employed cooperative redundancy elimination method. Routers periodically exchange the cache summary with one hop neighbors and create the Data graph and calculate the Minimum Dominating Set (MDS). According to MDS, the Data graph will then be assigned ranks and the one with the highest ranking is deleted.

For efficient caching in CCN, Cache Replacement is also an important factor. The Cache Replacement strategy creates free storage space for a new Data item when the cache of the router is full. Cache replacement strategies can be classified into five categories [8], recency-based, frequency-based, recency/frequency-based, function based and randomized strategies.

Another topic of interest in CCN is forwarding. If the requested data item is found within the CS of the current router, it is immediately returned to the user. Otherwise, the router forwards the Interest by Forwarding Strategy (FS). The original FS floods the network with the Interest through all the interfaces [1]. However, the flooding strategy leads to excessive processing of Interest packets for the neighboring routers. In [9] and [10], the authors proposed ranking of each interfaces by the router to choose the best path to retrieve the content. In [11], the authors introduced the Availability Info Base (AIB) to know which router stores which Data.

We focus on two main issues of CCN. Firstly, we try to reduce the caching redundancy by grouping routers. Secondly, we try to solve the scalability and load balancing in caching. We try to alleviate the unbalanced caching in the network caused by modulo hashing. We employ consistent hashing [15] to solve this problem. We categorize the routers into two groups as shown in Fig.1: Upper Layer Group (ULG) and Lower Layer Group (LLG). ULGs are connected with other ASs or LLGs whereas LLGs are connected with the users. Both ULGs and LLGs guarantee to store the chunks without overlap and forward the Interest directly to the corresponding routers. However, LLGs have to store the chunks based on the
We use consistent hashing to overcome the shortcomings of modulo hashing. The consistent hashing uses keys that are independent form the number of physical routers, as a modulus value. Here, keys are considered as a ring with values from (0 to n). In the original consistent hashing [15], a router is hashed and it is positioned according to its value as a point on the ring. The routers are randomly placed on the ring. Thus, when a router fails, it may cause load balancing issues; some remaining routers may have to handle large keys spaces whereas some only handle small keys spaces regardless of the physical cache size of the routers. To solve these issues, we place virtual routers like [16].

Virtual routers maintain the load balancing of the routers inside one group. One group of routers possess a key range (0 to n) and each key represents one virtual router which is a point on the ring. One physical router can possess several random keys or virtual routers depending on its physical cache size. For instance, a router 1 possess five keys and these keys are distribute randomly, for example, 0, 5, 8, 37, 90 and this router will cache the Data that has the hash values of 0, 5, 8, 37, 90. The router 2 possess three keys 1, 2, 15. In this way, when a router 1 is removed, the keys possessed by that router will not be dumped on a single router. By assigning the keys to each physical router within one group and a hash ring map is formed. The hash ring map is used to make the decision for forwarding Interest and caching Data. When a router receives an Interest, the Interest name is hashed. The hashed value is then matched with the hash ring to locate the router which is meant to have that requested Data. Then, Interest is directly sent to the dedicated router.

In case of a router failure, the point of the failed router is handled by the other existing routers in the same group. The incoming Interest and caching the Data on the failed router is mapped to the next highest point router. However, the key values mapping to the other router still remains the same. Similarly, addition of a new router is handled in the same fashion. Proposed Cache Decision and Forwarding Algorithms are discussed in detail in next sub sections.

C. Cache Decision

In our proposed cache decision, we divide an AS into several groups of routers as shown in Fig. 1. Cached Decision Algorithm (CDA) consists of two algorithms: CDA for ULG (CDAULG) and CDA for LLG (CDAALLG). An LLG router receives a content object with labeled chunks and caches only those chunks that Local Popularity (LP) is more than the threshold \( \alpha \). Here, we calculate LP of a chunk by using a counter for each interest in a router. When a cache miss occurs at the router, the value of the LP counter is increased by 1. If this counter crosses the \( \alpha \) value, then the respected data chunk is cached. Otherwise, the data chunk is not cached. If the group is the ULG, the router caches all the chunks without considering popularity. In CDA, Object \( O_i \) represents \( j \)th chunk of the content object and \( R^i_j \) represents the current router in \( j \) group. \( O_i(name) \) represents the content object name.

CDAULG: ULG router only checks whether the hash value of the Content chunk is for the current router or not. If the chunk is for the current router \( R^i_j \), it caches the chunk. At the same time, \( R^i_j \) forwards that chunk to requested faces by PIT list. If the the chunk is not for \( R^i_j \), it forwards the chunk to the corresponding router and other requested routers listed in PIT. This process is presented in Algorithm 1.

II. CONSISTENT HASHING BASED COOPERATIVE CACHING AND FORWARDING IN CCN

A. System Model

Autonomous System (AS) consists of two type of groups ULG(s) and LLG(s). We assume that each group contains equal number of routers with heterogeneous cache sizes. Each router possesses \( n \) keys (virtual routers) and caches the data depending on its keys. The keys are distributed, depending on the cache size of the router. The bigger the cache size of the router, the more keys it will possess. There are four types of routers, Negotiator Router (NR), Access Router (AR), Bridge Router (BR), Gateway Router (GR). NR collects the group information and construct the hash ring map. After constructing hash ring map, NR shares it with all other members inside the group. In this way, all the members know the keys range possessed by each router. AR connects with clients. BR connects one cluster with another cluster. GR connects one AS with another AS. At the initial stage, groups or clusters are manually formed by the administrator. The group formation depends on the location of the routers. Inside each group, each router is at most two hop distance from the NR. After this stage, the router, which has the highest betweenness centrality value, becomes the NR. Each router constructs the Neighbor Router Group Table (NRGT) which contains interfaces list to reach other routers in the group. NRGT consists of three fields: current router label, destination router label and interfaces. Finally, the hash ring is constructed and distributed by the NR.

B. Construction of Hash Ring with Virtual Routers

The main reasons for using the hashing is to eliminate the duplicate chunks in an on-line manner, to effectively forward the Interest and retrieve the Data. Modulo hashing is widely used for this purpose [12], [13], [14]. Although, modulo hashing is lightweight and efficient, it is not consistent and creates unbalanced loads. In modulo hashing, the routers in a group are used as buckets and modulo of the hash value is taken with respect to the total number of routers. The output value gives the designated router label to store the incoming chunk. Thus, when a content router is added or removed, the modulus value is also changed. Therefore, modulo hashing become inconsistent and most of the previously cached content objects’ locations are changed and these contents become useless. Solution for this problem is to delete all previously cached contents. Thus, the network can suffer the high fetching cost and traffic, even when only one router joins or leaves the group.

We use consistent hashing to overcome the shortcomings of modulo hashing. The consistent hashing uses keys that are independent form the number of physical routers, as a modulus value. Here, keys are considered as a ring with values from (0 to n). In the original consistent hashing [15], a router is hashed and it is positioned according to its value as a point on the ring. The routers are randomly placed on the ring. Thus, when a router fails, it may cause load balancing issues; some remaining routers may have to handle large keys spaces whereas some only handle small keys spaces regardless of the physical cache size of the routers. To solve these issues, we place virtual routers like [16].

Virtual routers maintain the load balancing of the routers inside one group. One group of routers possess \( n \) key range (0 to n) and each key represents one virtual router which is a point on the ring. One physical router can possess several random keys or virtual routers depending on its physical cache size. For instance, router 1 possess five keys and these keys are distribute randomly, for example, 0, 5, 8, 37, 90 and this router will cache the Data that has the hash values of 0, 5, 8, 37, 90. The router 2 possess three keys 1, 2, 15. In this way, when a router 1 is removed, the keys possessed by that router will not be dumped on a single router. By assigning the keys to each physical router within one group and a hash ring map is formed. The hash ring map is used to make the decision for forwarding Interest and caching Data. When a router receives an Interest, the Interest name is hashed. The hashed value is then matched with the hash ring to locate the router which is meant to have that requested Data. Then, Interest is directly sent to the dedicated router.

In case of a router failure, the point of the failed router is handled by the other existing routers in the same group. The incoming Interest and caching the Data on the failed router is mapped to the next highest point router. However, the key values mapping to the other router still remains the same. Similarly, addition of a new router is handled in the same fashion. Proposed Cache Decision and Forwarding Algorithms are discussed in detail in next sub sections.

C. Cache Decision

In our proposed cache decision, we divide an AS into several groups of routers as shown in Fig. 1. Cached Decision Algorithm (CDA) consists of two algorithms: CDA for ULG (CDAULG) and CDA for LLG (CDAALLG). An LLG router receives a content object with labeled chunks and caches only those chunks that Local Popularity (LP) is more than the threshold \( \alpha \). Here, we calculate LP of a chunk by using a counter for each interest in a router. When a cache miss occurs at the router, the value of the LP counter is increased by 1. If this counter crosses the \( \alpha \) value, then the respected data chunk is cached. Otherwise, the data chunk is not cached. If the group is the ULG, the router caches all the chunks without considering popularity. In CDA, Object \( O_i \) represents \( j \)th chunk of the content object and \( R^i_j \) represents the current router in \( j \) group. \( O_i(name) \) represents the content object name.

CDAULG: ULG router only checks whether the hash value of the Content chunk is for the current router or not. If the chunk is for the current router \( R^i_j \), it caches the chunk. At the same time, \( R^i_j \) forwards that chunk to requested faces by PIT list. If the the chunk is not for \( R^i_j \), it forwards the chunk to the corresponding router and other requested routers listed in PIT. This process is presented in Algorithm 1.
Algorithm 1 Cache Decision for ULG
1: Input: $O_i = \text{Content Object}$
2: hash $O_i(name)$ and match it in consistent hash map
3: if hash value of $O_i(name)$ is for the current router $R_C^i$ then
4: Cache the object $O_i$
5: Send object $O_i$ to the router(s) listed in PIT
6: else
7: Send object $O_i$ to the router(s) listed in PIT
8: end if

Algorithm 2 Cache Decision for LLG
1: Input: $O_i = \text{Content Object}$
2: hash $O_i(name)$ and match it in consistent hash map
3: if hash value of $O_i(name)$ is for the current router $R_C^i$ and $LP > \alpha$ then
4: Cache object $O_i$
5: Send object $O_i$ to the router(s) listed in PIT
6: else
7: Send object $O_i$ to the router(s) listed in PIT
8: end if

CDALLG: When an LLG router receives the Data chunks, it checks two conditions; hashed value of the chunk name and the LP value. If the chunk is for current router $R_C^i$ and also LP is greater than the $\alpha$, the router will cache the Content and forward the chunk to the requested face(s) listed in PIT. Otherwise, the data chunk is only forwarded on the requested face(s). This process is given in Algorithm 2.

D. Cooperative Forwarding
When the corresponding router receives the interest packet, there can be two situations. Firstly, the requested data is present in the corresponding routers CS. It will simply reply with the requested data chunk. Secondly, the corresponding router does not have the requested data chunk in its CS. In this situation, the corresponding router will send the request packet towards BR/GR and will enable the Direct Forward (DF) bit. We propose to add one bit to Interest packet header for DF (either 0 or 1). DF prevents the interest to be dropped by the intermediate routers.

For example, in Fig.2(b), user sends an Interest to router $R1$ whose label is 0. $R1$ hashes the Interest name to know the corresponding router. Let’s the hashed value of the Interest name is owned by $R_5$ whose label is 5. The Interest is forwarded to $R_5$ (label 5) thorough $R2$ (label 4). In here, $R2$ will not check the incoming Interest in CS. Lets suppose $R_5$ does not have the requested data in its CS. So, $R_5$ forwards the Interest to router $R2$ which acts as the BR and also as the NR. In this case, $R2$ will drop the Interest because the Interest is already listed in the PIT. To prevent this dropping, we need DF bit. In [12], the authors have solved this problem by using piggy Interest.

The cooperative forwarding is presented in Algorithm 3, where $I$ represents the Interest and $I'$ represents the DF enabled interest. $R_k'$ represents router $k$ and $R_j'$ represents current router in group $j$. $I(name)$ represents the Interest name.

Algorithm 3 Cooperative Interest Forwarding
1: Input: $I = \text{Interest}$
2: if $DF$ bit of $I = 1$ [i.e. updated/Direct Forward $I'$] then
3: Check the PIT list
4: if the Interest $I'$ is already in PIT then
5: Stop forwarding
6: else
7: Put $I'$ to PIT list and forward $I'$ to neighbor groups/AS via BR or GR
8: end if
9: else if $DF$ bit of $I = 0$ [i.e. normal $I$] then
10: hash $I(name)$ and match it in consistent hash map
11: if hash value of $I(name)$ is for current router $R_C^i$ then
12: Search $I$ in CS
13: if request object $O_i$ found in CS then
14: Reply $O_i$
15: else
16: update $I$ to $I'$ and Put $I'$ to PIT list
17: forward $I'$ to neighbor groups/AS via BR or GR
18: end if
19: else
20: Check PIT list
21: if Found then
22: Stop forwarding
23: else
24: Put $I$ to PIT list and forward $I$ to $R_k'$
25: end if
26: end if
27: end if

using ccnSim [17]. ccnSim is a chunk level simulator that is developed over Omnet++ simulator. We use homogeneous cluster size, heterogeneous cache size and different delays for simulation. Clients request the contents in a random manner governed by the zipf distribution. The performance of the proposed scheme and others are measured by success hit defined in eq 1. We used cache hit as the performance measurement which means the requested content is taken from the router cache. Server hit is the situation in which the requested content is not present in the routers’ cache and the request is forwarded to the original content provider. High cache hit means less traffic in the network and vice versa in case of server hit. The success hit is calculated by the following equation,

$$\text{Success cache hit} = \frac{\text{Cache hit}}{\text{Cache hit} + \text{Cache miss}}$$ (1)

We have constructed two LLGs and one ULG in the topology depicted in Fig.1. Each cluster or group contained 5 routers ($0, 1, 2, 3, 4$) and ARs connected with the users. In one cluster, the cache size of the router varies from 200 to 600 chunks and the total cache size of one group is 2000. The total cache size of the whole network is 6000 chunks. In simulating other algorithms, we do not construct the groups of routers. One content provider or repository contain 1000 contents and the average size of each content is 20 chunks. The content provider contains an average of 20000 chunks. Thus, the network can cache about 30% of the total contents. Clients request the contents in a random manner governed by the zipf distribution with different parameters $1, 1.2, 1.4, 1.6, 1.8$. One group of routers possess 100 keys.

In this simulation, we compared our proposed algorithm Consistent Hashing (CH) with others. For comparison, we use
TABLE I: Parameters used in the experiments

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>num repos</td>
<td>1</td>
</tr>
<tr>
<td>replicas</td>
<td>1</td>
</tr>
<tr>
<td>num clients</td>
<td>8</td>
</tr>
<tr>
<td>lambda</td>
<td>1</td>
</tr>
<tr>
<td>file size</td>
<td>20</td>
</tr>
<tr>
<td>key range</td>
<td>100</td>
</tr>
<tr>
<td>alpha</td>
<td>1, 1.2, 1.4, 1.6, 1.8</td>
</tr>
<tr>
<td>objects</td>
<td>1000</td>
</tr>
<tr>
<td>CDS</td>
<td>LCE, LCD, PROB</td>
</tr>
<tr>
<td>RS</td>
<td>LRU</td>
</tr>
<tr>
<td>FS</td>
<td>CH,NRR, NRR1, SPR</td>
</tr>
<tr>
<td>Cache</td>
<td>200, 300, 400, 500, 600</td>
</tr>
</tbody>
</table>

Fig. 3: Comparison of cache hit

(a) CH and \{NRR,NRR1,SPR\}PROB

(b) CH and \{NRR,NRR1,SPR\}LCD

(c) CH and \{NRR,NRR1,SPR\}LCE

a combination of the following algorithms. We simulate three forwarding decision algorithms (FS); Shortest Path Routing (SPR), Nearest Replica Routing (NRR), Nearest Neighbor Routing 1 (NRR1), with three cache decision algorithms (DS); LCE [1], LCD, and Probabilistic in network caching (PROB) [5]. The cache replacement algorithm is LRU.

Fig.3a, Fig.3b and Fig.3c display the average cache hit comparison of our proposed scheme CH with two thresholds (0 and 5) with other schemes. Fig. 3a show the average cache hit rate inside the network, CH is 21% better than NRR-PROB, 31% better than NRR1-PROB and 23% better than SPR-PROB. Similarly Fig.3b shows that cache hit rate of CH is 21% better than NRR-LCD, 31% better than NRR1-LCD and 22% better than SPR-LCD, respectively. The results shown in Fig.3c show that cache hit rate of CH is 28% better than NRR-LCE, 99% better than NRR1-LCE and 31% better than SPR-LCE, respectively.

IV. CONCLUSION

In this paper, we proposed the cooperative caching and forwarding schemes to keep the popular chunks inside the AS. We solved the scalability issue of the modulo hashing by employing consistent hashing. We also eliminated the load balancing issue of modulo hashing in this paper by using virtual router concept. We intensively simulated our proposed algorithms and our simulation results show that our proposal has a better cache hit ratio and lower sever hit ratio, thus leading to a decreased traffic inside the network. As a future work, we will use content popularity prediction for cache decision and cache replacement.

ACKNOWLEDGMENT

This research was supported by Basic Science Research Program through National Research Foundation of Korea(NRF) funded by the Ministry of Education (NRF-2014R1A2A2A01005900). Dr. CS Hong is the corresponding author.

REFERENCES