On Adaptive Pre-fetching and Caching the Contents in Content Centric Networking

Kyi Thar, Saeed Ullah, Doo Ho Lee, Choong Seon Hong

Department of Computer Science and Engineering, Kyung Hee University,
446-701, Republic of Korea
Email:{kyithar, saeed, dooholee, cshong}@khu.ac.kr

***Abstract*—In the initial proposal of Content Centric Networking (CCN), routers store all passing contents (a content composed of several segments). So, in the best case, the routers provide the contents directly to the users when the requested contents are stored in its cache. In the worst case, the requested content is just only located at the original server which is faraway from the current Autonomous System. Thus, the delay to get all the segments of requested content may be very high and users' Quality of Experience will be very low. In this paper, we propose a scheme to reduce the delay to get the contents, improve the cache hit and also reduce the hit distance. Core routers store only one copy of the contents and also forward the requests cooperatively. Then, as the main proposal the per-fetching algorithm is proposed at the access routers, in order to fetch the locally popular content(s) before requested by the users. We have intensively evaluated the performance of our proposed scheme by using the chunk-level simulator. We have shown in the simulation results that our proposal's performance is better than the other similar proposals.**

***Keywords*—Coordinated Caching, Prefetching, Content Centric Network, Cooperative Forwarding, Consistent Hashing.**

I. Introduction

According to the Cisco Visual Networking Index (VIN) [1], a demand of watching video through the Internet is increasing exponentially. The structure of current Internet, which is an end-to-end communication scenario, is not efficient enough to deal with this increasing demand. In order to cope with this traffic increase, Jacobson and et al. have proposed the Content Centric Networks (CCN) [2]. The main feature of the router used in CCN is to store contents temporarily and give the copy of stored content to the users when they request. CCN users request the content in the form of segmented sequences and these segments are call chunks. In CCN Interest packets are used to request the contents or find the contents inside the network and in a reply get the Data packet. One Interest packet retrieves only one Data packet which is one segment of a content or one chunk.

Whenever a router receives the Interest, first of all, it finds the requested chunk in the Content Store (CS), the cache memory of the router, that stores copies of the passing Data temporarily in the form of the chunk. If the requested chunk is found in its CS, the router immediately provides that chunk to the users and discards the Interest. Otherwise, the router checks its Pending Interest Table (PIT), which keeps the list of forwarded Interests. If the requested Interest is listed, updates the interface and wait for the returning chunk. Otherwise, the router checks its Forwarding Interest Base (FIB), which stored the lists of the incoming and outgoing Interface, in order to find the best path to forward the Interest. If there is no entry for the desired destination then the interest packet is flooded in the network through all the interfaces.

Originally, the forwarding strategy is, flood the Interests to find the chunk(s) inside the network. Then, all the returning chunks are stored at the CS of each router and that process is handled by the cache decision strategy. If the CS is full, the stored chunks are replaced with new chunks by using cache replacement strategy. Here, we would like to discuss several issues of CCN, especially on forwarding and caching strategy.

In the default CCN, a router floods the Interest to the neighbor routers to find the content. Flooding the Interests degrades the performance of the network because every router receives the Interests and need to go through its CS, PIT and/or FIB. Therefore, in [3], [4], router ranked the interfaces and choose best path to retrieve content objects. If the first ranked route doesn't provide the content, router chooses the second-ranked interface to forward that unsatisfied request. [5] uses Availability Info Base (AIB) to forward the Interests, however, it needs exchanging extra messages to update AIB. Classical hash based forwarding is used in [6] [7] [8] and it forwards request without periodically exchanging extra messages.

Originally, the routers on the requested path store the same contents which affect the utility of the cache space of the routers, this mechanism is called Leave Copy Everywhere (LCE). Leave Copy Down (LCD), which is proposed in [9] for the web caching, reduces the duplicate contents on the request path, but still don't eliminate it completely. Progressive Caching [10] extends the LCD to cache popular chunks and solve the problem of one timer cache in order to improve cache utilization of each router. In [11], the probabilistic caching scheme is proposed which takes the cache capacity of the path in consideration. According to this scheme, the routers closer to the users have higher probability for caching the content. [12] tries to solve the caching problem of the duplicated chunks around one hop neighbors by using cooperative redundancy elimination method. Router periodically exchange the information with one hop neighbors to eliminate the contents duplication that creates extra traffic in the network. Classical hashing is also used in [6] [7] [8] for caching purpose and it can reduce extra message exchanging to eliminate the duplicated contents and forwarding. Although, classical hashing is lightweight and efficient, but it suffers from consistency and load balancing problems.

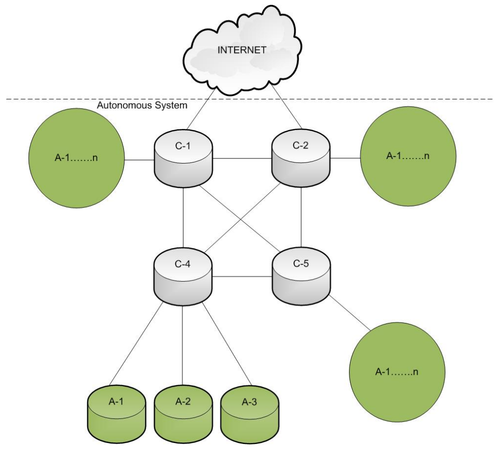

Fig. 1. System Model: where C is the core router and A is the access router. Core routers are connected with Internet/content server. The access routers are connected with users.

The prefetching technique is widely used in different areas. Several issues of the prefetching is clearly mentioned in [16] such as cache pollution problem and wasted Prefetches problem. Basically, there are two types of prefetching techniques, synchronous (prefetch the chunks when chunk i miss) and asynchronous (prefetch several sequence of chunks when hitting the chunk i).

In this paper, we propose a mechanism to improve the cache space utilization and also to reduce the delay to get the contents or chunks. There are two parts of Caching, forwarding and cache replacement schemes proposed in this paper: for Core Router (CR)s and for Access Routers (AR)s. For the case of CR, we use the scheme which is based on [13] [14], where consistent hashing is applied as a base frame for the caching and forwarding process of CR. Then we improve the CR's cache decision, in order to work together with the cache replacement, where the cache decision in [13] [14] works independently with cache replacement and it affects the performance and stability. As the main proposal for this paper, the prefetching scheme is proposed for AR, where prefetching technique improves the cache hit probability and it can provide better service to the users by fetching the contents before requested by users. As a result, CRs can keep non-duplicated contents in their cache and also it can forward the unsatisfying requests directly toward the *custodian routers*[1], instead of flooding the requests to find the contents. In addition, the latency to get the contents is also reduced because of prefetching technique.

Our contribution in this paper can be summarized as follows.

- The cache decision for CRs is proposed for working together with cache replacement.
- The adaptive prefetching algorithm for ARs is proposed, in order to improve the cache hit rate and reduce the latency to get the chunks.
- The proposed scheme is evaluated by using ccnSim [15].

[1]The predefined routers that store relevant contents and those contents are filtered by using consistent hash ring.

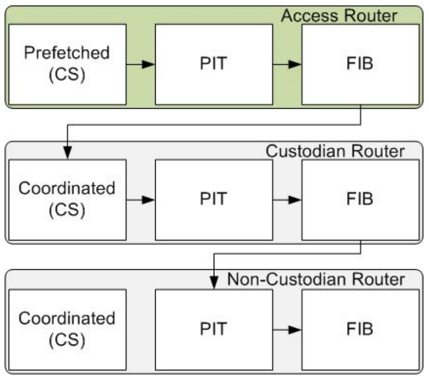

Fig. 2. The illustration of overview forwarding Process: If the router is not an CR or AR, the router will skip searching the contents in its CS.

Rest of the paper is organized as follow. Section II discusses system model while the overview process of the proposed scheme is explain in III. The cache decision process of core router is presented in IV. Adaptive prefetching algorithm been used in AR is discuss in section V-A. The proposed scheme is evaluated in the section VI. Finally the paper is concluded in section VII.

II. System Model

A system model of the proposed scheme is shown in Fig.1. The Core Routers (CR), Access Routers (AR) and users are located inside one Autonomous System (AS). The content servers are located at the outside of the AS. The white routers are the CRs (from $C_1, C_2, ...C_n$) and the green routers and circles are the ARs from $A_1, A_2, ...A_n$. Only the CR are grouped together by the system administrator and ARs are connected with CRs. The ARs are located in different geographical location and that are connected with CRs via fiber optic line(S). Users are connected with ARs and their requests first received by these ARs.

III. Overview Process

The overview process of the proposed scheme is as follow. CRs are formed as a group, possess the key range $(0, 1, 2...k)$ and each key represents one virtual router. One physical CR can possess several random keys or virtual routers, depending on the capacity of physical cache size. Several random keys are assigned to each physical core router and a consistent hash ring map is constructed. Then, Interest is forwarded directly to the custodian router by using consistent hash ring map.

The illustration of forwarding process can be found in Fig.2, where AR is the first router that receives Interest from users. Then the CR is categorized into two types of the router: Custodian and Non-Custodian CR. If the requested content is cached in AR, the content is replied directly to the user and also the prefetching Interest is forwarded to the CRs depending on the popularity of the satisfied chunks. If the requested content is not located at the AR, unsatisfying Interest and also the prefetching Interest(s) are forwarded to the CRs depending on the popularity of the unsatisfying chunk. When the unsatisfying or prefetching Interest arrives at the CR, if the CR is the Custodian then the requested content is searched in its CS. If the requested content is found, reply the chunk to the

AR. Otherwise, the Interest is forwarded to the another AS or origin server. If CR is not the Custodian, Interest is forwarded directly to the Custodian router. So, the benefit is that, if the CR is not a Custodian router, CR can skip the CS checking for the incoming Interest.

Algorithm 1 Core Router Cache Decision

```
1:  On arrival of the requested chunk at router, check the CB
2:  if CB is 0 then
3:      Drop incoming chunk to store and relay chunk to the requested router(s)
4:  end if
5:  if CB is 1 then
6:      if The cache space is free then
7:          Cache the chunk
8:      else {Hint : The cache space is full}
9:          Compare the value of the chunks
10:         if Incoming chunk < cached chunk then
11:             Drop incoming chunk to store and relay chunk to the requested router(s)
12:         else {Hint : Incoming chunk > cached chunk}
13:             Cache the chunk in CS and also relay the chunk to the requested routers
14:         end if
15:     end if
16: end if
```

IV. Core Router's Cache Decision Process

The cache decision process includes two steps. The first step is to know whether the current router is the custodian or not. If the current router is the custodian router, the router makes a decision (store or not store the chunk) by using the second step. The cache decision algorithm is shown in 1.

The first step is very simple, the router can know whether it is a custodian of the incoming Data chunks or not by checking Cache Bit (CB) field which is located in the PIT and CB value is updated by the forwarding process. If the CB value is 1, the incoming chunks are considered for caching by the router cache by using the second step. If the CB is 0, the incoming chunks will not be cached on current router cache.

For the second step, we combine the caching and replacement policy. There are two possible cases; the cache space of the CR is full or free. All the chunks that have CB value 1 are cached when the cache space of the CR is free. If the cache space of the CR is full, CR needs to replace the incoming new chunks with old chunks inside the cache. In order to do that, the router compares the value of incoming chunks and the chunks already cached in the cache. If the incoming chunk's value is higher than the chunks that are in the cache. Then, the incoming chunk is stored. At the same time, the old chunks in the cache are deleted. By this way, CRs eliminate the duplicated chunks without affecting the cached redundancy and effectively use the cache space.

V. Access Router Forwarding and Caching Process

The forwarding process of the ARs is very simple. Each AR just forwards the unsatisfying Interest toward the CR.

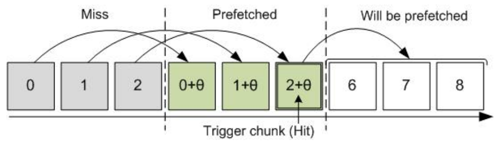

Fig. 3. Pre-fetching process: when the occurrence of the i (where $0, 1, ...$) chunks miss, router pre-fetch the $i + \theta$ (where $\theta = 3$) into its cache. If the trigger chunk hit occur, router pre-fetch the chunk sequence block depending on the trigger range.

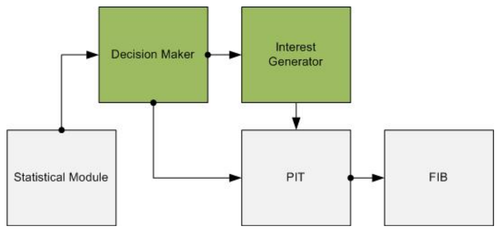

Fig. 4. The illustration of the Prefetching module(the green blocks) which includes Decision Maker and Interest Generator module.

Therefore, we will not discuss AR's forwarding as sub section. Major contribution in this section is the adaptive prefetching algorithm, which initially fetch the chunks one by one and latter, fetch the sequences of chunks (a group of chunks) when hitting the triggered chunks. The detailed discussion is given in section V-A.

First of all, we would like to discuss an overview of the prefetching technique. Pre-fetching is the technique to fetch some segments or some chunks of the content before been requested by the user(s). This technique can provide better service to the user(s) when they use delay sensitive application(s) (eg. watching video etc.). However, on the other hand, deployment of perfecting in the network can be faced with two major problems [16]. First one is the cache pollution problem, where it is defined as the case when the most useful chunks from the cache are replaced with pre-fetched chunks. Second problem is, the wasted pre-fetches, i.e., the case when the chunks are pre-fetched too early or too late and that lead the cache pollution and generates huge traffic inside the network.

Then, we propose the prefetching module which is shown in 4 for the AR and it includes decision maker module and Interest generator module. Decision maker module makes a decision whether to start the prefetching for the content or not by using the information from the statistical module and Interest generator generates the Interest to fetch the chunks before requested by the users. Statistical module collects the statistical information such as a number of hit and number of a miss for each chunk of the contents. The prefetching algorithm which is used in prefetching module will be discussed in section V-A.

A. *Adaptive Prefetching Algorithm for Access Router*

In this paper, the prefetching technique is only applied on the ARs, these are connected and located at the nearest place from the users. Instead of pre-fetching every chunk, our algorithm only fetches the popular contents, in order to reduce the cache pollution problem or to prevent the replacing of popular chunks with non-popular chunks. Every AR, is measuring local popularity by using the statistical module

Algorithm 2 Adaptive Pre-fetching and Cache Replacement

```
1: if # of $\sum req_i^n(t) > \sigma$ then
2:    Fetching process is started
3:    if chunk $i$ of content $n$ miss then
4:       request the $i$ and $i+\theta$ chunks
5:       when the requested chunk $i$ and $i+\theta$ chunks is arrived
6:       Relay the chuk $i$
7:       if $i$==$\theta - 1$ then
8:          Set $i+\theta$ chunk as $trigger\ chunk$
9:       end if
10:      if Cache space is free then
11:         Store the $i+\theta$ in the cache
12:      else
13:         the $i+\theta$ chunk is replaced with LRU position chunk
            and added at the MRU position
14:      end if
15:   end if
16:   if hitting the chunk $h$ (trigger chunk) then
17:      Request the sequence of chunks from $i+1$ to $i+g$
18:      where, $g$=$\lceil I_{ij}^n/RTT_i^n \rceil$
19:   end if
20: end if
```

(which keep track of the hitting and missing of contents). The pre-fetch range and trigger range, these terminology are widely used in the prefetching, also play an important role in pre-fetching. The wasted pre-fetch and cache pollution problems are solved by tweaking these values. The pre-fetch range is the range to fetch the chunks after cache miss occurs. The trigger range is the number to generate the Interest after cache hit occurs. The trigger range setting depends on the demand for the contents.

For example, in Fig.3, the pre-fetching range is defined as θ (where θ = 3), router will fetch the $i+\theta$ chunk where request of chunk i faces a miss. When the θ value is too high, some chunks may be deleted before assessing. By this way, the chunks are fetched on router cached before accessed by the users. Then, we apply the trigger chunks concept, where the occurrence of the hit on that trigger chunks, the router generates Interests to fetch the group of chunks, in order to balance the demand and response. The number of Interest to generate is depending on the value of g, which is the trigger range.

For instance, in the figure chunk number 5 (i.e.$2+\theta$) has been chosen as a trigger chunk because it is the last chunks for the prefetching. When hitting the trigger chunk, the router will generate g. The trigger range g is calculated according to equation 1.

The proposed adaptive prefetching and cache replacement is presented in in algorithm 2. In the algorithm, line 1 is to check whether the chunks is popular or not, where σ is the threshold (real number) and $\sum req_i^n(t)$ is the number of total request for the chunk i of content n. If the chunk is popular the fetching process is started for content n. Otherwise, just relay the chunk. When the fetching process is started, the router will request the chunk i and also chunk $i+\theta$ (e.g $i=0, \theta=3$). The router will store only the chunk $i+\theta$ and relay the chunk i to requested users. Line 7 shows how to choose the trigger chunk, where $i+m$ is set as trigger chunk. When trigger chunk hit occurs, the router will fetch the $i+1$ to $i+g$ (e.g. $i=6, g=3$). In the algorithm, g is calculated as

$$g = \lceil I_{ij}^n/RTT_i^n \rceil \quad (1)$$

where I_{ij}^n is the Interest arrival of content n and RTT_i^n is the round trip time to get chunk i of content n.

TABLE I. PARAMETERS USED IN THE EXPERIMENTS

Parameters	Symbol	Value
No. of repos	$\|r\|$	1
No. of replicas	$\|m\|$	1
No. of clients	$\|c\|$	9
Arrival rate	λ	100
file size	$\|F\|$	1
Zipf exponent	α	0.9,1,1.1
Object size	o_i	10^8
Cache decision	DS	CH(normal), Hybrid, LCE, LCD, PROB
Cache replacement	RS	Hybrid(deterministic), LRU
Forwarding	FS	CH,NRR , NRR1, SPR
Cache size	CS	5% of object size

TABLE II. STRATEGIES USED IN THE EXPERIMENTS

Strategies	Description
Leave Copy Everywhere (LCE)	Store all incoming chunk passing through the routers.
Leave Copy Down [17] (LCd)	Chunks will store at the down stream router of the hitting router.
Probabilistic Caching [11] (Prob-Cache)	Routers closer to the user have higher probability for caching the chunks.
Nearest Replica Routing(NRR)	Flooding Interest to the neighbor routers. If found reply the chunks. If the request time out, forward the Interest to the nearest content source.
Nearest Replica Routing(NRR1)	Flooding the Interest again and again. If the chunks is found, return to the requesters.
Least Recently Used (LRU)	Replaces the least recently used item with new incoming chunk.

VI. PERFORMANCE EVALUATION

In this section, we evaluate the proposed scheme by extending the chunk-level simulator, ccnSim [15] which is develop under Omnet++ simulator. Then, the proposed scheme is analyzed by deploying homogeneous group size, heterogeneous cache size, and the same delay. First, we discuss the configuration of the simulation environment. Secondly, we choose the performance metrics to compare the results of our proposed scheme with other. The final part is the discussion of the simulation results.

A. Configuration

The topology used in the simulation consists of four groups and each group located in the different geographic region. The core router group is connected with the outside of the network, in this scenario, it is connected with the content server. Three groups are connected with users. Then, we consider the catalog size of the contents that are kept at the content server, chose YouTube likes catalog size 10^8 which is also used in [18] [19] and the size of the contents on average are 10MB as in [20]. Also, the cache size of the router is assigned, heterogeneous cache size for the whole network with 0.1% and 5% of the catalog size is assigned. For the simplicity, we consider topologies with uniform delay (1ms). Then, we

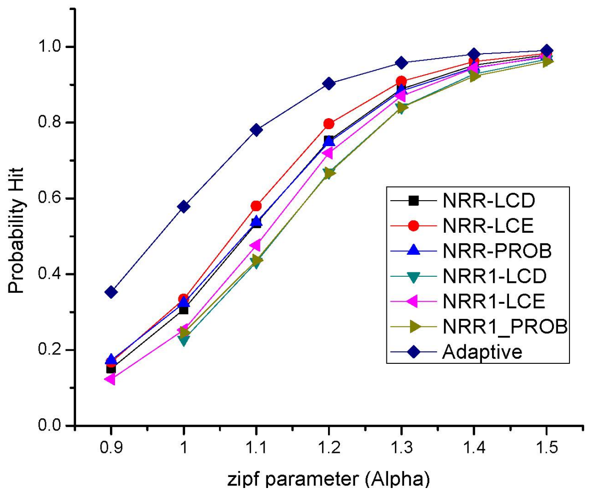

Fig. 5. Comparison of probability hit between proposed scheme (Hybrid) and others

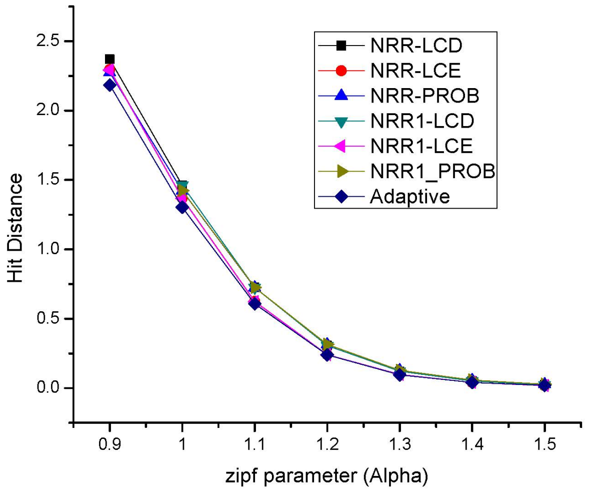

Fig. 6. Comparison of hit distance between proposed scheme (Hybrid) and others

assume the links are non-congested and with the infinite bandwidth. So, the network delay matches the propagation delay of each link. Object popularity follows the Zipf distribution probability equals $P(X = i) = \frac{1/i^{\alpha}}{c}$ with $C = \sum_{j=1}^{|F|} 1/j^{\alpha}$ where i is the rank of the i-th object, and we further denote $\Pi_i = \sum_{j=1}^{i} P(X = j)$ as the cumulative percentage of requests directed to the set of i most popular objects. We consider the object popularity follow a zipf distribution with (α=$0.9, 1, 1.1, 1.2, 1.3, 1.4, 1.5$). The requests arrival follow the Poisson process, $P\{N(t) = n\} = \frac{(\lambda t)^n}{n!} e^{-\lambda t}$. Simulation start from empty caches and statistics are gathered after the hit ratio reaches steady state.

B. Performance Metrics

CCN reduces the network traffic as well as the latency for the users. Typically, the performance of the cache is measured in terms of the probability that the chunks of Interest is found at a given CS. The probability of hit which measures the hit and miss probability in order to know how much traffic network can reduce. On the other hand, it can measure the server hit probability which represents the traffic pass through to the outside of the network. The probability of the hit is calculated by the following equation,

$$\delta = \frac{\nu}{\nu + \zeta} \tag{2}$$

,

where, δ is the probability of hit, ν is the total number of the hit for the whole network and ζ is the total number of the miss for the whole network. Also, the hit distance, the number of hops Interest travels before hitting a copy of the requested chunk, is also important to measure the performance. The hit distance roughly reflects the overall load on the network and the end users delay and also a user centric-metric, beyond the usual cache hit and miss probability, directly relates to users QoE (i.e., delay) as well as network QoS (i.e., load and cache hit). The average distance is calculated by the following equation,

$$d_{avg} = \frac{\sum m_i * d + h}{\sum m_i + 1} \tag{3}$$

,

$$\text{D} = \frac{D_{avg}}{|c|} \tag{4}$$

. where, d_{avg} is the average distance or average number of hops to get the contents,

In addition, measure the latency to download the contents from the user side and it is calculated by using the following equation,

$$\kappa = \varphi * \frac{1}{|c|} \tag{5}$$

, where, φ is the global average delay and $|c|$ is the number of clients. Finally, to measure the performance of our proposed scheme, we choose five performance metrics (probability of hit, hit distance, inner hit, average delay, and cache diversity).

C. Simulation Results

Network Centric Performance : Fig.(5) shows CCN performance in the chunk level hit rate. As we expected, hit rate of all schemes are increased when we increase the Zipf parameter α from (0.9 to 1.5). In the figure, our proposed scheme clearly outperforms the others because of adaptive pre-fetching, which requests the content before receiving the user(s) request. LCE and LCD results are lower than ours because these policies cache the content after the cache miss occurs.

Client Centric Performance Hit distance: Fig.(6). shows hit distance measurement, i.e., how many hop the request need to forward to satisfy the request. The lower the hit distance the better the performance. The figure shows hit distances for all the schemes are almost the same.

Average delay: The comparison of average delay is shown in fig.(7). Average delay shows a kind of round trip time from sending Interest and receiving Data. In this figure, the average

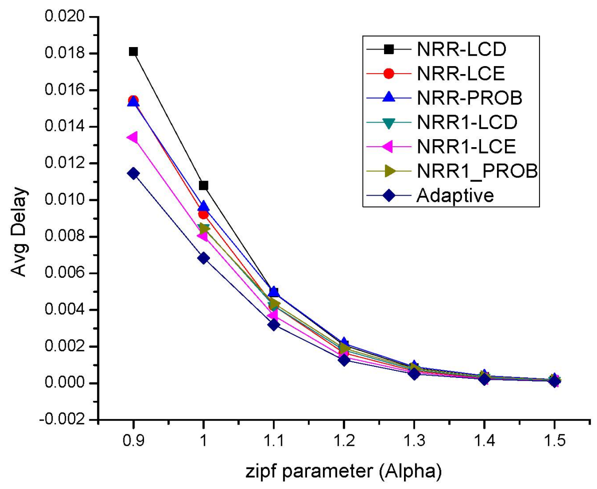

Fig. 7. Comparison of average delay between proposed scheme (Hybrid) and others

delay to get the data of our proposed scheme is outperforming other because of the pre-fetching technique.

VII. Conclusion

In this paper, we proposed a scheme to store the contents with the combination of cooperative and adaptive prefetching. Aim of the proposal in this paper is to enable the user to get the requested content quickly. We have used a modified version of LRU in the core routers for content replacement in order to give more priority to the popular contents to be cached. Furthermore, the proposed forwarding scheme forwards the requests directly to the custodian router without flooding the requests. As a result, the proposed scheme can improve not only the cache hit but also the hit distance and average delay to download the contents. We intensively simulated the proposed mechanism. The experimental results show that for our proposed mechanism, the cache hit is higher and the average waiting time to get the content is lower than other similar proposals in the literature.

Acknowledgment

This research was funded by the MSIP(Ministry of Science, ICT & Future Planning), Korea in the ICT R&D Program 2015. *Dr. CS Hong is the corresponding author

References

[1] http://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/index.html.

[2] Van Jacobson, Diana K Smetters, James D Thornton, Michael F Plass, Nicholas H Briggs, and Rebecca L Braynard. Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 1–12. ACM, 2009.

[3] Cheng Yi, Alexander Afanasyev, Lan Wang, Beichuan Zhang, and Lixia Zhang. Adaptive forwarding in named data networking. *ACM SIGCOMM computer communication review*, 42(3):62–67, 2012.

[4] Michele Tortelli, Luigi Alfredo Grieco, Gennaro Boggia, and K Pentikousisy. Cobra: Lean intra-domain routing in ndn. In *Consumer Communications and Networking Conference (CCNC), 2014 IEEE 11th*, pages 839–844. IEEE, 2014.

[5] Shuo Guo, Haiyong Xie, and Guangyu Shi. Collaborative forwarding and caching in content centric networks. In *NETWORKING 2012*, pages 41–55. Springer, 2012.

[6] Zhe Li and Gwendal Simon. Time-shifted tv in content centric networks: The case for cooperative in-network caching. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–6. IEEE, 2011.

[7] Lorenzo Saino, Ioannis Psaras, and George Pavlou. Hash-routing schemes for information centric networking. In *Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*, pages 27–32. ACM, 2013.

[8] Sen Wang, Jun Bi, and Jianping Wu. Collaborative caching based on hash-routing for information-centric networking. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 535–536. ACM, 2013.

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

[10] Jason Min Wang and Brahim Bensaou. Progressive caching in ccn. In *Global Communications Conference (GLOBECOM), 2012 IEEE*, pages 2727–2732. IEEE, 2012.

[11] Ioannis Psaras, Wei Koong Chai, and George Pavlou. Probabilistic in-network caching for information-centric networks. In *Proceedings of the second edition of the ICN workshop on Information-centric networking*, pages 55–60. ACM, 2012.

[12] Jason Min Wang, Jun Zhang, and Brahim Bensaou. Intra-as cooperative caching for content-centric networks. In *Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*, pages 61–66. ACM, 2013.

[13] Kyi Thar, Saeed Ullah, and Choong Seon Hong. Consistent hashing based cooperative caching and forwarding in content centric network. In *Network Operations and Management Symposium (APNOMS), 2014 16th Asia-Pacific*, pages 1–4. IEEE, 2014.

[14] Kyi Thar, Thant Zin Oo, Chuan Pham, Saeed Ullah, Doo Ho Lee, and Choong Seon Hong. Efficient forwarding and popularity based caching for content centric network. In *Information Networking (ICOIN), 2015 International Conference on*, pages 330–335. IEEE, 2015.

[15] Giuseppe Rossini and D Rossi. ccnsim: an highly scalable ccn simulator. In *IEEE ICC*, 2013.

[16] Binny S Gill and Luis Angel D Bathen. Amp: Adaptive multi-stream prefetching in a shared cache. In *FAST*, volume 7, pages 185–198, 2007.

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

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

[19] Giuseppe Rossini and Dario Rossi. A dive into the caching performance of content centric networking. In *Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2012 IEEE 17th International Workshop on*, pages 105–109. IEEE, 2012.

[20] Phillipa Gill, Martin Arlitt, Zongpeng Li, and Anirban Mahanti. Youtube traffic characterization: a view from the edge. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 15–28. ACM, 2007.