Pre-fetching Scheme for Content Centric Networking

Kyi Thar, Jae Hyeok Son, Choong Seon Hong
Department of Computer Science and Engineering, Kyung Hee University
{kyithar, sonjaehyeok, cshong}@khu.ac.kr

Abstract

In the original Content Centric Networking (CCN), routers are attached with cache storage, in order to provide the contents instantly to the users when the requested contents are already stored in router's cache. Otherwise, routers search the requested contents inside the network. For the best case, contents can be acquired from one hop neighbors. For the worst case, requested contents is downloaded from the server, a faraway location and this situation affect the user's QoS. In order to improve the QoS and to get the contents with low latency, we propose a pre-fetching scheme, which fetch the content before the arrival of user's requests, and it is deployed at the access router. For the core routers forwarding and caching, Consistent Hashing based forwarding and caching is applied. Then, we evaluated the performance of our proposed scheme by using ccnSim. On average, our scheme is outperform 14% than the SPR-LCE.

1. Introduction

According to the Cisco Visual Networking Index (VIN), the demand of watching video through the Internet is increasing very rapidly. Content-Centric Networking (CCN) [1] is proposed by Jacobson and et al, in order to cope with the increasing traffic. CCN users request the content in the form of segmented sequences and these segments are call chunks. Router uses Interest packets to request the contents and in a reply, get the Data packet. One Interest packet retrieves only one chunk.

In this paper, we propose a mechanism to improve the cache space usage and performance of the network. For the Core Router (CR)'s forwarding and caching process, we invite the reader to read [3] to understand the processes. As the main proposal, the prefetching technique is proposed and deployed at the Access Router (AR) to improve cache hit and to reduce the latency. Thus, the router can provide better service by fetching the contents, before been requested by the users. As a result, CRs keep non-duplicated contents and forward the unsatisfying requests directly toward the custodian routers (the predefined routers to store relevant contents by filtering consistent hash ring), instead of flooding the requests. Besides, the latency to get the contents are reduced because of prefetching technique.

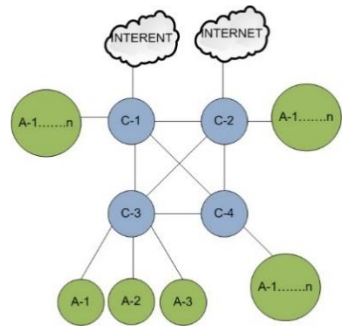

Figure 1 System Model

2. System Model

A system model of the proposed scheme is shown in Fig.1. All the users and routers are located inside one Autonomous System (AS). The content servers are located outside of the AS. The blue circles are the CRs and the green circles are the ARs. The CRs 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.

Algorithm 1 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 $i+m$ chunks
5:     when the requested chunk $i$ and $i+m$ chunks is arrived
6:     Relay the chuk $i$
7:     if Cache space is free then
8:       Store the $i+m$ in the cache
9:     else
10:      the $i+m$ chunk is replaced with LRU position chunk and added at the MRU position
11:    end if
12:  end if
13:  if hitting the chunk $i$ (already fetched) then
14:    Request the $i+m$ chunks
15:  end if
16: end if
```

3. Core Router Forwarding and Cache Decision Process

This section discusses the overview process of CR's forwarding and caching. The detail process can be found in [3]. Each CR possesses several keys to store the contents and forward the Interest. One of the CR becomes the Negotiator Router (NR) and collects the keys information. Then, the keys are formed as a key value ring and that is distributed to all of the CRs by NR. When the Interest arrives at the CR, the Interest's name is hashed and the output value is compared with CR's own keys. If values are the same, CR is the custodian and it searches the requested content in its CS. If it is found, CR replies the content. Otherwise, Cache Bit (CB) is set as 1 and forwards to the server. If the output values are different, CB bit is set as 0 and Interest is forwarded to the custodian router. For the cache decision, if the incoming content's CB value is 1, then CR stores the content and also relays the content. If the CB value is 0, CR just relays the content. As a result, the cache utilization is improved by storing only one copy of the content inside the group of CRs.

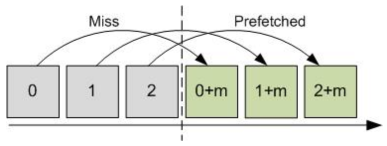

Figure 2 Pre-fetching Process

4. Access Router Forwarding and Pre-fetching Process

AR forwarding is very simple, just forwards the Interest toward the CRs. The main contribution in this section is the adaptive prefetching algorithm. The pre-fetching is the technique to fetch 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.). The chunks are fetched when the chunks miss and also the pre- fetched chunks hit.

Our algorithm only pre-fetches the popular chunks, instead of fetching every chunk, in order to reduce the cache pollution problem [3]. The pre-fetching range is defined as m, which is important for pre-fetching process and it solves the wasted pre-fetch problem [3] by tweaking these values. When the m value is too high, some chunks may be deleted before assessing. In Fig.2, the pre-fetching range is m (where $m = 3$). So, the router will fetch $i + m$ chunk, when chunk i miss. Also, when the occurrence of hit on the chunks that are already fetched (the pre-fetched portion /green potion), the AR will pre-fetch the next $i+m$ chunks. The proposed adaptive pre-fetching and cache replacement algorithm is shown in Algorithm 1. In this algorithm, line 1 is to check the condition, whether the chunks is popular or not, where σ is the threshold (real number) and $\sum req_i^n(t)$ is the total number of requests for the chunk i of content n. If the chunk is popular, the fetching process is started for content n. Otherwise, AR just relays the chunk and does not store on its cache. When the fetching process is started, the router will request the chunk i and also chunk $i+m$ $(e.g\ i = 0, m = 3)$. The router will store only the chunk $i+m$ and relay the chunk i to users. If hitting the chunk i which is already fetched in the cache, then the router will request the next $i+m$ chunks. For the cache replacement, we modified the LRU policy.

5. Performance Evaluation

We extend the ccnSim [4] to run experiments. For the simulation topology, there are 4 CRs and 4 ARs are attached with CRs. The parameter used in the experiments are shown in Table 1. The probability of hit and average delay, as a performance metric to measures the hit/miss probability and average delay to get the chunks. The strategies used in experiments are shown in Table 2.

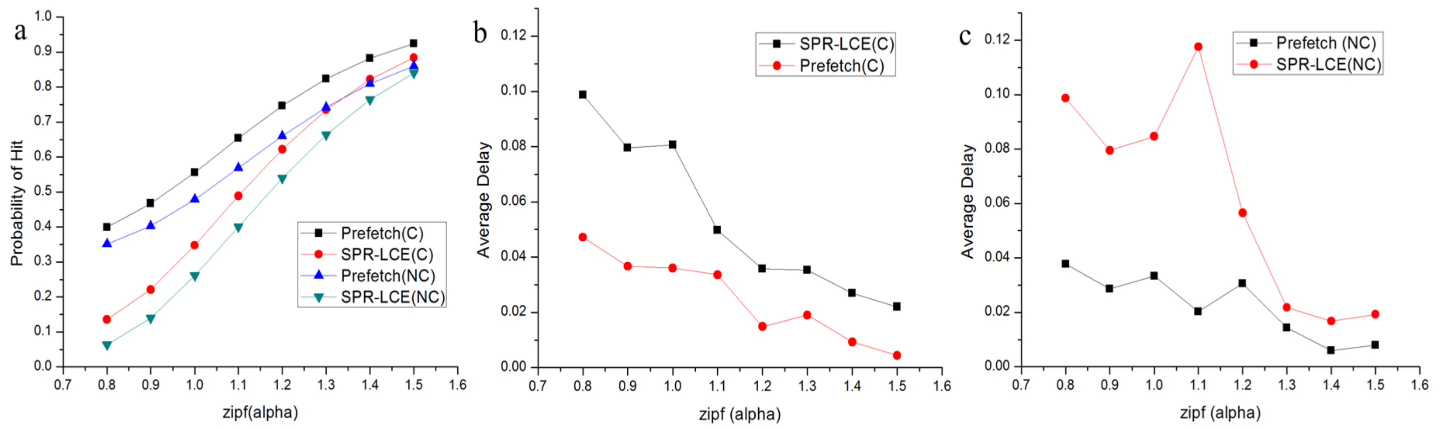

Figure 2 Comparison of proposed scheme with others: (a) Probability of cache hit, (b) Average delay (CRs with cache), (c) Average delay (CRs without cache)

For the experiments, Pre-fetch (C) and SPR-LCE (C) are simulated with CRs have cache capacity to store the contents. Pre-fetch (NC) and SPR-LCE (NC) are simulated with CRs do not have cache capacity to store the contents. Fig. 2(a) shows the comparison of a cache hit between proposed prefetching scheme and SPR-LCE, where we run the experiments with different zipf distribution parameter (alpha) values from 0.8 (the contents are similarly popular) to 1.5 (the few contents are so much popular). In the results, ours prefetching scheme is outperformed than the SPR-LCE, either CRs have the cache to store the contents or without cache. This is because, ARs fetch the chunks on its cache before been requested by users. On average, our scheme’s performance is outperform 14% (CRs with cache) and 15% (CRs without cache) than the SPR-LCE. Fig.2 (b) and (c) shows the comparison of the average delay, where our proposed scheme serves the contents with lower delay, either the CRs can store the contents or not. On average, our proposed scheme is outperform 2.8% (CRs with cache) and 4% (CRs without cache) than the SPR-LCE.

6. Conclusion

In this paper, we proposed a prefetching scheme which can improve the user’s QoS especially for delay sensitive application such as Video on Demand (VOD). The experimental results show that the proposed scheme achieved the better performance than the others. For future work, we will analyze the proposed scheme with large network scenario.

Table 1 Parameters used in the experiments

Parameters	Symbol	Value
Chunk size	c	10KB
No. of repos	$\|r\|$	1
No. of replicas	$\|m\|$	1
No. of clients	$\|c\|$	4
Arrival rate	λ	100
file size	$\|F\|$	20
Zipf exponent	α	0.8 t0 1.5
Object size	o_i	20^4
Cache decision	DS	LCE,Prefetching
Cache replacement	RS	LRU
Forwarding	FS	SPR
Cache size	CS	5% of the catalog size

Table 2 Strategies used in the experiments

Leave Copy Everywhere (LCE)	Store all incoming chunk passing through the CRs.
Shortest Path Routing (SPR)	CRs chose the shortest path to server and send Interest.
Least Recently Used (LRU)	CRs replace the least recently used chunk with new chunk.

Acknowledgement

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] Jacobson, Van, et al. "Networking named content." international conference on Emerging networking experiments and technologies. ACM, 2009

[2] Thar, Kyi, et al. "Efficient forwarding and popularity based caching for Content Centric Network." IEEE International Conference on Information Networking (ICOIN), 2015.

[3] 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.

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