Applying Multi-Armed Bandit Problem into Cache Decision Algorithm for Content Centric Networking

Kyi Thar, Tuan LeAnh, and Choong Seon Hong
Department of Computer Science and Engineering, Kyung Hee University.
Email: {kyithar, latuan, cshong}@khu.ac.kr

Abstract

Among the future Internet architecture, Content Centric Networking (CCN) is one of the most promising network architectures, where Content Routers (CRs) are attached with the cache space to store the contents temporary and CRs provide the contents to users from their cache. The original caching scheme stores every arriving contents and that manner leads to the fast cache replacement. So, the popular contents are replaced with non-popular contents. Thus, we proposed a cache decision algorithm, which stores only popular contents and also prevents the replacing of popular contents with non-popular contents. We have intensively simulated the proposed mechanism in a chunk level simulator and the performance is compared with existing schemes. The simulation results show that the proposed mechanism outperforms the existing state-of-the-art schemes.

1. Introduction

In Content Centric Networking (CCN) architecture, Content Routers (CRs) are attached with the cache to store Data packets (chunk) temporarily to satisfy requests (Interest packets) in near future [1]. Originally, the CRs store all the arriving chunks. Thus, the cache replacement is so fast, and the popular contents are replaced with non-popular contents. Therefore, the researchers proposed different schemes to solve this problem [2], [3], [4]. So, in this paper, we proposed a cache decision algorithm based on the multi-armed bandit problem, to improve the cache utility, where it prevents the replacing of popular contents with non-popular contents.

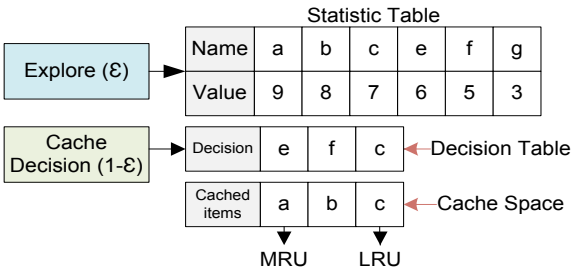

Figure 1 Cache decision process

Fig. 1 shows the overview process of proposed cache decision scheme. The proposed scheme consist of two phases: 1) exploration and 2) exploitation. Exploration phase learns and updates the value of each content items. Exploitation phase, which is also known as cache decision phase, produces the content items list to store or replace with the cached items, where the CR replace the cached items with the Least Recently Use (LRU) policy. The LRU policy manages two pointers to point out the Most Recently Used (MRU) items and LRU items. By using LRU policy, the new content item is stored at the MRU position and the content at the LRU position will be deleted. Every time period, the CR chooses the exploration phase with probability ϵ, and exploitation phase with probability $1-\epsilon$.

2. Multi-Armed bandit Problem (MAP) preliminaries

In MAP, a gambler makes a decision to choose the slot machines (also known as multi-armed bandits) to play, in order to maximize the rewards. In this problem, each machine has an unknown reward. Some machine can give a lot of rewards and some are not. In the environment of classical MAP, the gambler can only choose one machine par round but in this paper, he chooses several machines per round.

3. System Model

There is total N number of CRs, and each CR is n. Then, the CRs are attached with the cache space, and the size of cache space is s_n, where $\sum_{n=1}^{N} s_n = S$. The decision table on each CR which helps the cache decision is denoted as d_n. The table to store the statistical information is denoted as c_n. The content o of chunk i is denoted as o_i, where $o_i \in O$. The value of the each content o at time t is v_o^t. The cache miss and hit of each content o is m_o^t and h_i^t. The reward of each content o at time t is r_o^t, and the value of content o at time t is v_o^t.

Algorithm 1 Decision List Updating

1: Every time t
2: If $\delta > \varepsilon$ then
3: Choose exploitation phase
4: $d_n \leftarrow \emptyset$
5: $d_n \leftarrow$add the items name with the descending order of its value, until the decision list is full
6: Else
7: Choose exploration phase
8: $r_o^t \leftarrow (h_{o=1}^{O} / \sum_{o=1}^{O} h_i^t) * r_i^{t-1}$
9: $v_o^t \leftarrow (m_o^t / \sum_{o=1}^{O} m_i^t) * v_o^{t-1} + r_i^t$
10: $c_n \leftarrow v_o^t$
11: Sort c_n values in descending order
12: End If

4. MAP based cache decision algorithm

In this paper, a gambler corresponds to a CR, rewards corresponds to the cache hit, and each machine corresponds to content items in the decision list. The size of this decision list is depending on the cache capacity of each CR. Every time t, the router chooses exploration or exploitation phase depending on the ϵ. Every exploration phase, each CR updates the contents values and stores these values in the statistical data list c_n with descending order. For every exploitation phase, each CR updates their decision list d_n, which included the contents items name to store in its cache. Then, the decision list d_n is used a filter for final cache decision.

Whenever, Data packets or chunk are arriving at the router, it checks the decision list. If the arriving contents are listed in the decision list, CR stores the content. Two cases can occur, the cache space is free and the cache space is full. When the cache space is free, the CR stores the content directly in its cache at the MRU position. Otherwise, the CR removes the LRU position items and store the new content at the MRU position.

Algorithm 1 shows the decision list updating process and Algorithm 2 shows the cache decision process. In the Algorithm 1, every time t, CR can choose two possibilities, explore or exploit. If the δ (Bernoulli random distribution) is greater than the ε, CR chooses the exploitation phase (line 2 and 3), where ε is the fixed value between 0 and 1. Otherwise, CR chooses the exploration phase. Every exploitation phase, CR first clears the decision list, then fills up the decision list with the contents by descending order of the

Algorithm 2 Cache Decision

1: On the arrival of the chunk d_i
2: If $o_i \in d_n$ then
3: If $\sum_{i=1}^{k} d_i < s_n$ then
4: Store the chunk d_i at the MRU position
5: Else
6: Remove the LRU position chunk and store the new chunk d_i at the MRU position
7: End If
8: Else
9: Do not consider to store the new chunk d_i
10: End If

content values until the list is full (line 5). Every exploration phase, CR updates the value and sorts the items in descending order. Finally, CR updates the ε (line 11 and 12).

In the Algorithm 2, on the arrival of the chunk d_i, CR first checks the decision list d_n. If the arriving chunk d_i is listed, then the router considers caching (line 2). Otherwise, CR does not consider to store the arriving chunk d_i.

5. Performance Evaluation

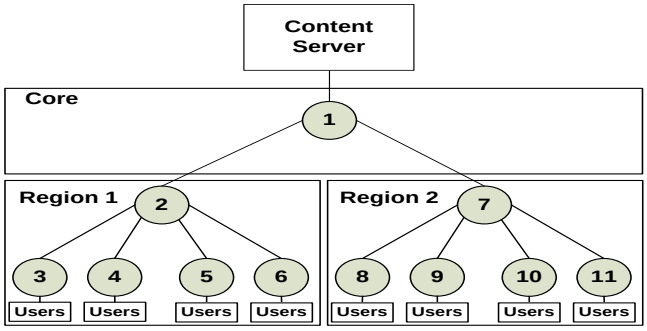

Figure 2 Topology used in experiment

In this section, we present the evaluation of proposed mechanism, and simulate the proposed scheme using the chunk-level simulator, ccnsim [5] which is developed under Omnet++ simulator. For the experiment, the cluster size of the network is homogeneous and the cache capacity of each router is heterogeneous. Table 1 shows the parameters use in the experiments. Clients request the contents in a random manner governed by the Zipf distribution. We chose cache hit probability and inner cache hit ratio to compare the performance of the proposed scheme with others. Fig. 2 shows the topology uses in simulation. In this simulation, there is one content server, and that contains 1000 contents. The average

size of each content is 10 MB. Thus, the content provider contains average 10000 MB contents. The network can cache up to 5% of the total contents. Then, we compare our proposed scheme with others existing schemes (three forwarding decision algorithms and three cache decision algorithms). For the forwarding decision algorithm (FS), we choose Nearest Neighbor Routing (NRR1). For the cache replacement, we choose LRU. For the cache decision algorithm we choose three cache decision algorithms (DS) to compare with proposed scheme. These are, 1) LCE [1], 2) LCDand3) PROB.

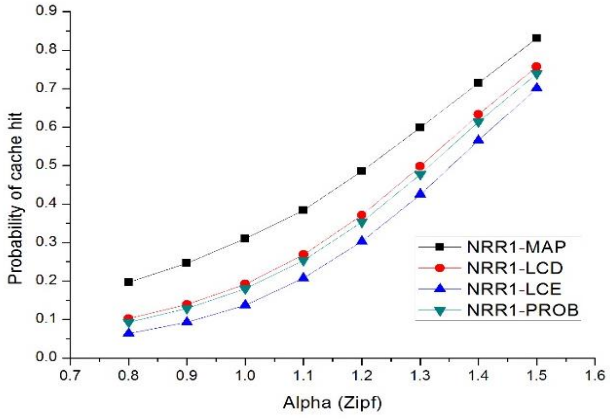

Figure 3 Comparison of probability of cache hit

Fig.3 shows the cache hit comparison between the proposed scheme and the others. The results show that the proposed scheme outperforms than others because the proposed scheme prevents the replacing of popular contents with non-popular contents.

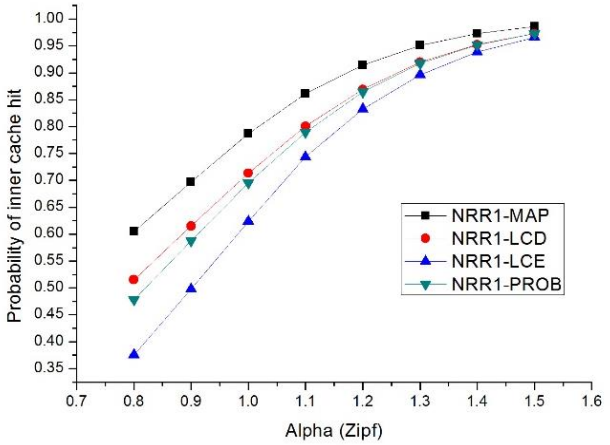

Figure 4 Comparison of inner cache hit

Fig. 4 shows the inner cache hit comparison between proposed scheme and the others, where the proposed can reduce the outgoing traffic than others.

Parameters	Values
Number of Content Server	1
Number of clients	11
λ (user arrival)	1
α (Zipf)	0.8-1.5
Catalog size	20^4
Average file size	10KB
t	1 Sec
ϵ (0< ϵ<1)	0.5
Total cache size	5% of the catalog size

5. Conclusion

In this paper, we proposed a caching scheme which is based on Multi-armed bandit problem. We intensively simulated the proposed mechanism. The experimental results show that according to our proposed mechanism, the cache hit is higher than other proposals in the literature.

6. Acknowledgement

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (B0101-16-0033, Research and Development of 5G Mobile Communications Technologies using CCN-based Multi-dimensional Scalability) *Dr. CS Hong is the corresponding author.

7. References

[1] 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.

[2] 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

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

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

[5] Chiocchetti, R., Rossi, D., & Rossini, G. (2013, June). ccnsim: An highly scalable ccn simulator. In IEEE International Conference on Communications (ICC), 2013 (pages. 2309-2314).