Hash-Based Cache Distribution and Search Schemes in Content-Centric Networking

Yurino SATO††(a), Member, Yusuke ITO††(b), Student Member, and Hiroyuki KOGA††(c), Member

SUMMARY Content-centric networking (CCN) promises efficient content delivery services with in-network caching. However, it cannot utilize cached chunks near users if they are not on the shortest path to the server, and it tends to mostly cache highly popular chunks in a domain. This degrades cache efficiency in obtaining various contents in CCN. Therefore, we propose hash-based cache distribution and search schemes to obtain various contents from nearby nodes and evaluate the effectiveness of this approach through simulation.

key words: cache distribution, hash function, CCN, ICN

1. Introduction

Content-centric networking (CCN) is designed for content delivery services—which are one of the most popular Internet services—and it has attracted much attention [1]. In CCN, content is segmented into smaller chunks that are stored in content routers (CRs) during forwarding. A user requesting content sends interest packets to the server along the shortest path (i.e., default path). The default path is constructed by the forwarding information base (FIB) on each CR. A CR that receives the interest packets sends the requested chunks back to the user instead of the server through the reverse path traversed by interest packets if it caches them; otherwise, it forwards the interest packets to upstream CRs or the server. Through these procedures, users can efficiently obtain chunks from nearby CRs in CCN. However, chunks cached in nearby CRs that are not on the default path cannot be retrieved. And, each CR tends to mostly cache highly popular chunks because frequently requested chunks are often duplicated. This will degrade cache efficiency in obtaining various contents in CCN.

Some approaches have been proposed to solve these problems. One approach is to distribute and obtain chunks to/from all CRs in a domain based on hash-routing techniques [2]. To obtain chunks, this approach creates the shortest path to a responsible CR based on a hash function, which maps chunk identifiers to CRs to be cached. A different approach, called breadcrumbs [3], has been proposed. In this approach, CRs create and manage a pointer that indicates the direction in which CRs might cache chunks based on the forwarding history to obtain chunks from the indicated CRs not on the default path. In both approaches, when the responsible CR does not store the requested chunks, it forwards the interest packets to the server. Another approach is to use flooding techniques to expand the search range near users [4]. A larger search range can improve the possibility of obtaining chunks from nearby CRs, although it will increase the overhead. These approaches can effectively obtain chunks from CRs regardless of whether they are on the default path or not. However, there are some problems: (1) the time needed to obtain content may increase due to a roundabout path when a responsible CR does not store requested chunks, and (2) a large search range will increase the overhead.

We propose more efficient hash-based cache distribution and search schemes to obtain various contents from nearby CRs with a high degree of certainty in CCN. Our scheme distributes cached chunks to each responsible CR by using a hash function, and it searches them by dynamically updating FIB entries. We evaluate the effectiveness of the proposed scheme.

2. Proposed Scheme

Our scheme distributes and obtains cached chunks to/from all CRs in a domain that is managed and supervised by a single entity or organization like autonomous system (AS), in order to efficiently obtain various contents from nearby CRs even if they are not on the default path. To distribute cached chunks over the domain, the proposed scheme uses a hash function that maps chunk identifiers to responsible CRs. When a CR receives assigned chunks (i.e., hash values of received chunks match its ID), it caches the received chunks with priority. To do this, the cache space of CRs is divided into two spaces: a priority cache space to store assigned chunks with priority and a non-priority cache space to store other chunks in an original manner. CRs that newly cache assigned chunks advertise its information (newly cached) to all neighbor CRs by flooding to update FIB entries. Similarly, CRs that discard assigned chunks advertise its information (discarded) to all neighbor CRs to update FIB entries. Through these temporary FIB entry updates, our scheme can search distributed chunks from nearby CRs not...
on the default path. Our scheme limits the maximum size of priority cache space and advertising range to suppress the overhead of flooding and FIB entry updates.

Here, we describe the operation of our scheme. First, hash values are assigned to each CR in a domain in advance (i.e., CR’s identifiers). Note that the performance of our scheme will be affected by the way to assign CR’s identifiers, although this issue will be addressed in future work to focus on fundamental characteristics of it in this study. Next, to obtain content, a client sends interest packets to the server through the default path. A CR that receives the interest packets sends the requested chunks back to the client instead of the server through the reverse path traversed by interest packets if it stores them; otherwise, it forwards the interest packets to neighbor CRs or the server. The interest packets are forwarded to nearby CRs if they have notified that the requested chunks are stored as assigned chunks through FIB entry updating (as mentioned later); otherwise, they are forwarded to the server through the default path. Namely, the proposed scheme executes the same operation to obtain content as original CCN. Next, we explain how to cache chunks proposed scheme executes the same operation to obtain content, a client sends interest packets to the server to change the number of CRs in the network. In this simulation, CR’s identifiers (hash values) were assigned to each CR from the lower left to the upper right in ascending order as shown in Fig. 1 (b). Hash values of chunks were calculated as a remainder of dividing the sum of content’s identifier (0–199) and chunk’s identifier (0–63) of each chunk by the total number of CRs. The priority rate of cache space on CRs is defined as the ratio of priority cache space to all cache space; e.g., the rate of 0.8 means that the priority cache space is 800 chunks and the non-priority cache space is 200 chunks. Each client repeatedly requested content at
equal a CR’s identifier, the received chunks are stored in a priority cache space, or in a non-priority cache space, otherwise. If the cache space to be stored is full, chunks stored in the cache space are discarded according to cache discard algorithms such as least recently used (LRU), as shown in Fig. 1 (b).

3. Simulation Model
To investigate the efficiency of our scheme, we evaluated it using Network Simulator ns-3 ver. 3.24.1 after implementing our scheme. The simulation parameters are summarized in Table 1. We used a simple grid topology that had multiple paths, like Fig. 1 (b), so that we focus on the fundamental characteristics of our scheme. The grid size was varied to change the number of CRs in the network. In this simulation, CR’s identifiers (hash values) were assigned to each CR from the lower left to the upper right in ascending order as shown in Fig. 1 (b). Hash values of chunks were calculated as a remainder of dividing the sum of content’s identifier (0–199) and chunk’s identifier (0–63) of each chunk by the total number of CRs. The priority rate of cache space on CRs is defined as the ratio of priority cache space to all cache space; e.g., the rate of 0.8 means that the priority cache space is 800 chunks and the non-priority cache space is 200 chunks. Each client repeatedly requested content at

*Table 1 - Simulation parameters.*

| Parameter                        | Value                  |
|----------------------------------|------------------------|
| Grid size                        | 4–12                   |
| Bandwidth of each link           | 1 [Gb/s]               |
| Delay time of each link          | 5 [ms]                 |
| Content size                     | 192 [chunk]            |
| Chunk size                       | 3 [Kbyte]              |
| Cache size on CRs                | 1000 [chunk]           |
| Priority rate of cache space     | 0–0.8                  |
| Cache algorithm                  | LRU                    |
| Advertising range of flooding    | 4 [hop]                |
| Number of contents               | 200                    |
| Zipf α                           | 0.8                    |
| Simulation time                  | 35 [s]                 |

*Fig. 1 - Proposed scheme.*

*Fig. 2 - Effect of priority rate of cache space.*
uniform random intervals with a range from 0.0 to 0.1 s and rate of requesting each content (content popularity) followed a Zipf distribution [5].

In this simulation, in order to focus on the basic efficiency of our scheme to distribute and obtain cached chunks to/from all CRs in a domain, we assumed that packet loss would not occur. Performance of our scheme was evaluated and compared to that of the conventional schemes (original CCN and hash-routing) by focusing on the average number of hops needed to obtain content, FIB entries, and FIB entry updates. The performance is not affected by link characteristics, so that a bandwidth and a delay time of each link were respectively set to 1 Gb/s and 5 ms, assuming a typical link. To focus on the performance in steady state, it was evaluated from 5 to 35 s after the simulation starts to avoid the influence of a transient period.

4. Simulation Results

Figure 2 shows the average number of hops needed to obtain content for the proposed scheme and conventional schemes (original CCN and hash-routing) when the priority rate of cache space for the proposed scheme varies from 0 to 0.8. Here, the grid size is set to 4.

When the priority rate is too small (< 0.05), the number of hops of the proposed scheme is almost the same as that of the conventional schemes. This is because the priority cache space is frequently updated, so the cache hit ratio decreases, and chunks are likely to be obtained from the server rather than CRs. On the other hand, when the priority rate is larger than 0.05, the proposed scheme drastically improves the number of hops. However, when the priority rate is too large (> 0.5), the number of hops increases. In this situation, the cache hit ratio on priority cache space increases, while that on non-priority cache space decreases. As a result, chunks are likely to be obtained from responsible CRs (which are relatively far away from the client) as priority cached chunks rather than nearby CRs as non-priority cached chunks.

Figure 3 shows the average number of hops, FIB entries, and FIB entry updates when the grid size varies from 4 to 12. Our scheme drastically improves the number of hops needed to obtain content regardless of the grid size, while the conventional schemes increase the number of hops in proportion to the grid size. Regarding network overhead, the number of FIB entries and FIB entry updates decreases due to a smaller number of assigned chunks per CR as the grid size increases. In a range of small grid size, a larger priority rate causes a larger number of FIB entries and smaller number of FIB entry updates, while a smaller priority rate causes a larger number of FIB entry updates and a smaller number of FIB entries. This is because a larger cache space on each CR stores a larger number of assigned chunks as well as is less frequently updated. Consequently, the proposed scheme with an appropriate priority rate can improve the number of hops needed to obtain content. The priority rate should be set to high if the flooding overhead of FIB entry updates should be reduced, while it should be set to low if the lookup overhead of FIB entries should be reduced.

5. Conclusion

We proposed efficient hash-based cache distribution and search schemes to obtain various contents from nearby CRs in CCN even if they were not on the default path. This scheme distributes cached chunks to each responsible CR in a domain by using a hash function and searches them by dynamically updating FIB entries. Simulation evaluations have indicated that the proposed scheme improves the number of hops needed to obtain content. In our future work, we will evaluate the characteristics of our scheme in detail as well as consider the most effective way to distribute and obtain cached chunks to/from CRs in more practical environment.

Acknowledgments

This work was supported in part by JSPS KAKENHI Grant Number 16H02806.

References

[1] V. Jacobson, D.K. Smetters, J.D. Thornton, M.F. Plass, N.H. Briggs, and R.L. Braynard, “Networking named content,” Proc. ACM CoNEXT2009, pp.1–12, Dec. 2009. DOI:10.1145/1658939.1658941
[2] L. Saino, I. Psaras, and G. Pavlou, “Hash-routing schemes for information centric networking,” Proc. ACM SIGCOMM ICN2013, pp.27–32, Aug. 2013. DOI:10.1145/2491224.2491232
[3] E.J. Rosensweig and J. Kurose, “Breadcrumbs: Efficient, best-effort content location in cache networks,” Proc. IEEE INFOCOM2009, pp.2631–2635, April 2009. DOI:10.1109/INFCOM.2009.5062201

[4] Y. Sato, T. Kawano, and H. Koga, “An efficient content search scheme to expand the search range in content-centric networking,” Proc. IEEE DRCN2015, pp.81–82, March 2015. DOI:10.1109/DRCN.2015.7148988

[5] C. Fricker, P. Robert, J. Roberts, and N. Shihi, “Impact of traffic mix on caching performance in a content-centric network,” Proc. IEEE NOMEN2012, pp.310–315, March 2012. DOI:10.1109/INFCOMW.2012.6193511