An In-Network Caching Strategy for Reducing Playback Interruption Time of On-Demand Streaming over CCN

Takeshi Misu¹, Masaki Hanada¹, Hidehiro Kanemitsu² and Moo Wan Kim¹

¹ Department of Information Systems
Tokyo University of Information Sciences
4-1 Onaridai, Wakaba-ku, Chiba 265-8501, Japan
E-mail: {g15008tm@edu, mhanada@rsch}.tuis.ac.jp

² Global Education Center, Waseda University
1-6-1 Nishiwaseda, Shinjuku-ku, Tokyo 169-8505, Japan

Abstract

Recently, Content-Centric Networking (CCN) has emerged as a new networking paradigm. The main features of CCN architecture are in-network caching and content-based routing. Cache replacement and decision policies have been discussed as important mechanisms of in-network caching in past literature. However, these policies cannot efficiently support streaming. In this paper, we propose an in-network caching method for reducing playback interruption time of on-demand streaming over CCN through efficient use of network resources. The proposed method splits streaming content data into chunks with fixed size and stores these chunks into the cache storage of each CCN router in order to reduce playback interruption time for streaming. Experimental evaluations show that the proposed in-network caching method outperforms in-network caching methods of the original CCN architecture in terms of playback interruption time.

1. Introduction

Recently, Content-Centric Networking (CCN) has emerged as a new networking paradigm [1]. The main features of CCN architecture are in-network caching and content-based routing. The aim of in-network caching is to keep recent content and frequently used content as long as possible because they are expected to be required again in the near future. Consequently, compared with distant content-servers, in-network caching responds quickly to content requests, improving network performance in areas such as bandwidth usage and delay. In most in-network caching applications, content files are sliced into chunks, and cache operations occur at the unit of a chunk. Cache replacement and decision policies have been discussed as important mechanisms of in-network caching in past literature. The cache replacement policy determines which content is to be evicted, and the cache decision policy determines which content is to be placed at which cache routers. In terms of cache replacement policies, LRU (Least Recently Used), LFU (Least Frequently Used), and FIFO (First In Fast Out) were proposed and analyzed for various traditional systems such as the Web, Content Delivery Networks (CDN), and Peer-to-peer (P2P) cache systems [2]. Additionally, cache decision policies such as LCE (Leave Copy Everywhere), LCD (Leave Copy Down) and MCD (Move Copy Down) have recently been proposed for CCN because cache routers on CCN architecture are not fixed [2]. Although various cache replacement and decision policies have been proposed, these policies cannot efficiently support streaming, which allows users to play video chunks (content chunks) while they are downloading them. The main reason is that the cache replacement and decision policies proposed in the past do not consider playback sequences of content chunks.

In this paper, we propose an in-network caching method for reducing playback interruption time for streaming over CCN. In the proposed method, the chunks of a content file are placed at CCN routers in the order of the chunks’ playback sequence starting at the router nearest to the user. Using experimental evaluation, we show that the proposed in-network caching method outperforms traditional in-network caching methods in terms of playback interruption time.

2. Streaming over CCN

In this section, we describe the model of streaming over CCN assumed in this study, as shown in Fig1.

In the CCN architecture, an Interest message, used to request content, is forwarded by each CCN router along the path toward the content-server. The content files are split into chunks with fixed size, and the user requests the content by chunks from the beginning of the content file. A Data message, supplying actual data, is returned by each CCN router
along the reverse path of the Interest message. The data is stored in the cache storage of the intermediate CCN routers through which the Data message is routed. In more detail, when a CCN router receives an Interest message, a longest-match lookup is performed on the Content Store (CS), Pending Interest Table and Forwarding Information Base (FIB), respectively. If a matching Interest message is found in the CS, the content will be sent out at the arrival face. If a matching Interest message is found in the FIB, it is forwarded according to the FIB.

Users (hereinafter referred to as consumers) play chunks from the beginning of the content file after finishing downloading the chunks. If playback of a chunk reaches a point where the chunk has not yet been downloaded, it must wait for the chunk to be downloaded [3]. Let us denote the $i$-th chunk by $C_i$ ($i = 1, \ldots, N$). Ideally, chunk $i$ should be downloaded before chunk $i - 1$ has finished playing so that playback is not interrupted. Hereinafter, the playback finishing time of chunk $i - 1$ is referred to as the playback deadline of chunk $i$. The playback interruption time, the playback waiting time, and the playback deadline are shown in Fig.2.

Figure 2: Playback interruption time and playback deadline

In Fig.2, because chunk 2 does not meet its playback deadline, the playback interruption of chunk 2 occurs. In order to reduce the playback interruption time for streaming over CCN, efficient distributed placement of each chunk that precedes the playback deadline is needed.

3. The Proposed In-Network Caching Method

The proposed in-network caching method consists of an initial placement phase followed by a replacement phase. During the initial replacement phase, a consumer requests the content file for the first time. Additional requests of the same content file by the same consumer are processed in the replacement phase.

The basic ideas of the proposed in-network caching method are as follows: In the initial placement phase, chunks from earlier parts of the content file are placed at CCN routers (hereinafter referred to as nodes) near the consumer; chunks from later parts of the content file are placed at nodes farther from the consumer (i.e., nearer to the content-server). To realize this mechanism, the proposed method adopts a priority-based strategy. For nodes near to the consumer, this strategy assigns high priority to chunks with a smaller playback sequence number, and it assigns low priority to chunks with a larger playback sequence number. Conversely, at nodes far from the consumer, chunks with smaller playback sequence numbers receive lower priorities than chunks with larger sequence numbers. If the cache storage is full, the proposed method replaces low priority chunks with the high priority chunks. In the replacement phase, the proposed method uses statistical transmission delay times of Data messages to judge whether each chunk can meet its estimated playback deadline. If a chunk is judged as likely to miss its estimated playback deadline, then that chunk is copied to cache storage in nodes that are able to meet the deadline. The initial placement phase and the replacement phase of the proposed in-network caching method are described in detail as follows:

**Initial Placement Phase:**

**Step 1:** When a consumer requests content chunks according to their sequence numbers for the first time, the consumer sends an Interest message ($IM_i$) for each chunk ($C_i$) toward the content-server.

**Step 2:** When the content-server (or the node which stores the $i$-th chunk ($C_i$)) receives the Interest message ($IM_i$) for the $i$-th chunk ($C_i$), a Data message ($DM_i$) is returned along the reverse path of the Interest message ($IM_i$). Here, the content-server (or the node which stores the $i$-th chunk ($C_i$)) inserts the sequence number of the chunks ($i$) and the total number of chunks ($N$) into the Data message header.

**Step 3:** When the $h$-th node ($CR_{h,i}$) receives the Data message ($DM_i$), the priority ($D_{h,i}$) of the $i$-th chunk ($C_i$) is calculated by using the algorithm shown in Fig.3. The $i$-th chunk ($C_i$) is stored in the cache storage of the $h$-th node ($CR_{h,i}$) if the cache storage is not full. If the cache storage is full and there exists a chunk whose priority is lower than the priority ($D_{h,i}$) of the $i$-th chunk ($C_i$), then the chunk with the lowest priority is replaced by the received $i$-th chunk ($C_i$). In the algorithm of Fig.3, $H$ is the total hop count and $w$ is the weight parameter.

An example of chunks stored on each node is depicted in Fig.4, assuming the cache capacity of the nodes is 5, the total number of chunks ($N$) is 9, the total hop count ($H$) is 3, and the parameter ($w$) is 0.6. In the figure, the consumer resides at node 0, and the content-server resides at node 4. In the initial placement phase, node 1 (i.e., the nearest node to the consumer) stores chunks 1, 2, 3, 4, and 5; node 2 (i.e., the middle node) stores chunks 3, 4, 5, 6, and 7; and node 3 (i.e., the nearest node to the content-server) stores 3, 4, 5, 6, and 7.

Transmission delays between nodes of Data messages are used for determining potential playback deadline miss. The calculation of transmission delay times occurs at each node when it receives a Data message. In this calculation, described next, $T(CR_{h,i}, CR_{h+1,i})$ denotes the actual transmission delay of Data message ($DM_i$) from node $CR_{h,i}$ to node $CR_{h,i}$, and $\overline{T}(CR_h, CR_{h+1})$ denotes a statistical trans-
mission delay estimate for recent transmissions from node $CR_h$ to node $CR_{h+1}$.

**Transmission Delay Calculation:**

When a node (including the content-server) sends a Data message ($DM_i$), the node inserts the send time into the Data message header. Similarly, when a node receives a Data message ($DM_i$), the node inserts the receive time into the Data message header. From the send and receive times included in a Data message ($DM_i$) header, the consumer continually updates the statistical transmission delay time via the assignment $T(CR_h, CR_{h+1}) = (1 - \alpha)T(CR_h, CR_{h+1}) + \alpha T(CR_{h,1}, CR_{h+1,1})$. Here, $\alpha$ is parameter that determines how fast we forget the past.

**Replacement Phase:**

**Step 1:** When a consumer requests a content chunk ($C_i$) according to its sequence number $i$ for the second time (or later), the estimated playback deadline ($ED_i$) is calculated and inserted into the Interest message header. $ED_i$ is given by $G + P \times R \times (i - 1) + I_1$ where $R$ is the playback bit rate, $P$ (bytes) is the size of the chunk, $I_1$ is the statistical transmission delay from the node which has the first chunk ($C_1$) to the consumer, and $G$ is the time at which the consumer sends the Interest message. In addition, the consumer inserts all of the statistical transmission delay times (i.e., the $T(CR_h, CR_{h+1})$) computed by the transmission delay calculation) into the Interest message header.

**Step 2:** When the $h$-th node ($CR_h$) receives the Interest message ($IM_i$) and the requested chunk ($C_i$) is hit, the $h$-th node judges whether or not the chunk ($C_i$) can meet its estimated playback deadline ($EA_i$). $EA_i$ is given by $V + \sum_{k=h}^{i} T(CR_{k-1}, CR_k)$ where $V$ is the time at which the node receives the Interest message. If $ED_i < EA_i$ is satisfied, the $h$-th node judges that playback deadline miss occurred, so the chunk ($C_i$) is copied to the node to meet its estimated playback deadline. In the copy destination node, the priority ($D_{i,h}$) of the $i$-th chunk is calculated using the algorithm shown in Fig. 3. If the cache storage of the $h$-th node ($CR_h$) is full, the chunk with the lowest priority is removed from the cache storage.

An example of information included in the Interest message header is depicted in Fig. 5. An example in which a playback deadline miss occurs is depicted in Fig. 6. In both examples, the cache capacity of nodes is 5, the total number of chunks ($N$) is 9, the total hop count ($H$) is 3 and the parameter ($w$) is 0.6.

In Fig. 6, the playback deadline miss of chunk 8 occurred at node 3. In order to meet the estimated playback deadline, chunk 8 is copied to the cache storage of node 2. The lower part of this figure describes the playback deadline judgment from the replacement phase for each node.
4. Evaluation by Experimental Simulation

In our experimental evaluation, we used the ndnSIM simulator implemented on an NS-3 architecture to compare playback interruption times of the proposed in-network caching method with traditional in-network caching. The experiment parameters are shown in Table 1, and the network topology [4] used in this simulation is shown in Fig.7. We assume that one producer is attached to each leaf node, depicted by the open circles in Fig.7; and 50 consumers are attached to each of the other leaf nodes, depicted by closed circles.

![Figure 7: Network topology](image)

Table 1: Experiment parameters

| Parameter          | Value |
|--------------------|-------|
| Bandwidth          | 1 Mbps |
| Playback Bit Rate  | 4 Mbps |
| Chunk Count (per a content file) | 1000 |
| Chunk Size         | 1 Mbyte |
| Node Count         | 40 |
| Content File Count | 25 |
| Cache Capacity     | 600 (chunks) |
| Parameter $\omega$ | 0.6 |
| Parameter $\alpha$ | 0.5 |

Table 2: Experiment results

| Method       | Average Playback Interruption Time |
|--------------|-----------------------------------|
| Proposed     | 0.1005 (sec)                      |
| LFU          | 0.9885 (sec)                      |
| LRU          | 0.9585 (sec)                      |
| FIFO         | 1.1099 (sec)                      |

![Figure 8: Histogram for the LRU and proposed method](image)

To start, we randomly selected one content file from 25 different content files without duplication, and the selected content file was stored in a randomly selected producer. Each consumer then randomly selects one content file and requests the file (i.e., 1000 chunks) twice with a random interval between 0 and 10. Table 2 shows the average interruption time under the proposed method, LFU, LRU, and FIFO. For the following discussion, we compared the proposed method with LFU because the average interruption time under LFU was the smallest among LFU, LRU, and FIFO. The upper and lower histograms of Fig.8 show the average interruption time per content file (i.e., 1000 chunks) under the proposed method and under LFU, respectively. These histograms do not include interruptions of zero seconds. Out of 900 requests, the proposed method had 833 zero-second interruptions, while LFU had 536 zero-second interruptions. Fig.9 and Fig.10 show the cache hit ratios under the proposed method and under LFU, respectively. The x-axes of Fig.9 and Fig.10 show the hop distance between a consumer and the node on which the consumer’s chunk request is hit.

![Figure 9: Cache hit ratio (Proposed method)](image)

![Figure 10: Cache hit ratio (LFU)](image)

From Table 2, the average interruption time under the proposed method is the smallest among all considered methods. This is because the frequency of a zero-second interruption under the proposed method is much higher than the other methods. In addition, comparing Fig.9 and Fig.10, we see that the proposed method has a high occurrence of smaller hop distances between a consumer and the node that provides a requested chunk. In other words, compared with LFU, the proposed method has more chunk requests that are hit at nodes near the requesting consumers. From these results, the proposed method better supports smooth real-time playback on streaming over CCN, as compared with traditional in-network caching methods.

5. Conclusions

In this paper, we propose an in-network caching method for reducing playback interruption time for streaming over CCN. In the proposed method, the chunks of a content file are placed at CCN routers in the order of the chunks’ playback sequence starting at the router nearest to the user. In the experimental evaluation, we showed that the proposed in-network caching method outperforms traditional in-network caching methods. Future work includes the evaluation under various system parameters.

Acknowledgment

This work is supported by JSPS KAKENHI Grant Number 16K00159 (Principal Investigator: Masaki Hanada) and the corresponding author is Masaki Hanada.

References

[1] V. Jacobson et al: Networking named content, ACM CoNEXT., pp. 1-12, 2009.
[2] G. Zhang et al: Caching in information centric networking: A survey, Elsevier Computer Networks., pp. 3128-3141, 2013.
[3] A. Vlavianos et al: BiToS: Enhancing BitTorrent for supporting streaming applications, Proc. INFOCOM’06, pp. 1-6, 2006.
[4] H. Li et al: Two-level popularity-oriented cache replacement policy for video delivery over CCN, IEICE Trans. Commun., Vol.E-99B, No.12, pp. 2532-2540, 2016.