Cache Replacement Strategies for Scalable Video Streaming in CCN

Junghwan Lee, Kyubo Lim, and Chuck Yoo
Dept. Computer Science and Engineering
Korea University
Seoul, Korea
{jhlee, kblim, chuck}@os.korea.ac.kr

Abstract—To deal with the structural limitations of the current Internet, a new concept of a future Internet has been suggested, among which a content centric network(CCN) with in-router storage and the use of content as an address has taken center stage. We describe the design and implementation of new cache-replacement algorithms for layered video content(H.264/SVC) streaming in a CCN. The cache-replacement algorithms(LGF-size, and Lmix) proposed in this paper impose value on each object based on its size, recent reference trend, and the reference frequency stored in the cache. In addition, the proposed algorithms operate by reflecting the layered features of SVC video streaming. The proposed algorithms are applied to a CCN node, and when a client repeatedly requests HTTP adaptive video-streaming encoded with H.264/SVC, the existing algorithms(FIFO, LFU, LRU) and cache hit rates of each CCN node are measured. A comparison of the cache hit rate at each CCN node and the quality of the video streaming, show that the efficiency of layered video streaming in a CCN can be improved by modifying the cache-replacement algorithm used.

I. INTRODUCTION

As a current trend, Internet users are moving toward another direction in terms of their content consumption. In addition to simple communication, users are tending to use Internet enabled communication devices to transmit their desired content and share data. Therefore, new types of communication have been suggested based on user trends. A content centric network(CCN) [1] is composed of an address system based on the name of the content, and is replacing the existing host based address system. Each CCN node has content storage where data packets are stored when passing through, which reduces duplicated data transmissions that may occur in the network and reduces the length of the data transmission route, allowing a faster response time to be secured.

On the other hand, the volume of data used in video streaming on the Internet has tended to increase dramatically. According to the Cisco Visual Networking Index(CVNI) [2], Internet video data will make up approximately 54% of all internet traffic, and the volume of data is expected to increase gradually. Therefore, if the structure of the future Internet includes an in-network cache, such as in a CCN, and reflects a specialized policy for video streaming, Quality of Service(QoS) of the client can be improved owing to a reduction of network traffic and a fast response speed. Scalable Video Coding(SVC) video content is an attractive factor to achieve this. If SVC is applied to the original video data, the efficiency of the video streaming can be improved when extracting the features of the video at various qualities from the generated data. More specifically, SVC data are featured with layered data internally. If all layered data are separated and used for video transmission in the CCN, the reuse rates of the in-network cache in the CCN will be increased, which will improve the video transmission efficiency [3]. We suggests and implements a new cache replacement policy for higher cache reuse rates in a CCN environment and compare with the existing cache replacement policy. An improvement of the cache hit rate and a low level of cache duplication of each CCN node are shown. The Layered Greedy-Dual size(LGD-size), and Layered Mix(Lmix) algorithms, which reflect the weighted values for each content object of the cache replacement policy based on the layer information of the SVC video data packet, were therefore designed and implemented.

II. RELATED WORK

A. Content-Centric Network

CCN is a Named Data Networking(NDN), which is a communication structure based on an address system using the data name. For a CCN packet, the name of the content is used as the address of a packet. In a CCN, both the interest packet and data packet types are all used. The interest packet requests a data packet from a CCN node, whereas data packet is a response to an interest packet and stores and transmits the actual content of the packet. The forwarding engines of a CCN node can be divided into three parts; a Forwarding Information Base(FIB), Content Store, and a Pending Interest Table(PIT). Each CCN node stores the Content Store in the data packet that the node has transferred itself. When the interest packet for a stored data packet is entered into a CCN node, Content Store immediately transfers the data packet that it has been storing. On the contrary, the interest packet for a data packet that is not stored in the CCN node is entered, and it only broadcasts the interest packet to the nearby CCN node. Afterward, when it receives a data packet for an interest packet, it saves the packet in the Content Store and transfers it to the interface where a interest packet is received.

Figure 1 shows the operating scenarios of a CCN node related with the Content Store. The scenario marked in white is one in which requested contents are stored in the Content Store of a CCN node, whereas the scenario marked in grey is one in which the requested contents are not stored in the Content Store of a CCN node.
B. Scalable Video Coding

H.264/SVC [4] is scalable part of H.264/AVC and the output is added to the standard related to the scalability of video data to H.264 standard. When video is encoded and decoded using SVC, the output of the video encoding can be separated into various layers. Encoding files contain not only base layers with essential information, but also an enhancement layer with information used for quality improvement. To decode a video image, data from base layer to the enhancement layer at a respectable bitrate are needed. This means that the data of the upper layer cannot be used without data from the lower layer, and therefore the upper layer is dependent on the lower layer. Figure 2 shows the decoding process when SVC data composed of one base layer and three enhancement layers are used. To use the lowest bitrate video, the base layer data must be decoded using the SVC decoder. To use a higher quality video image, decoding is conducted using not only the information from the enhancement layer but also data from each lower layer.

C. Cache-Replacement Algorithm

1) Greedy-dual(GD)-size algorithm:

\[ \text{Value}(i) = 1/i \]

GD-size algorithm [5], the initial value \( (i) \) is set using formula 1. When objects in the cache need to be deleted, the object with least value will be deleted, and the values of remaining objects will be decreased by the value of the deleted object. Through this calculation, a non-referred object stored in the cache is continuously decreased in value, and is likely to be deleted in the following process. If the stored object is referred again, the value is set to the initial value. This calculation increases the value again, and reduces the possibility of a future deletion. The time complexity of GD-size algorithm is \( O(\log n) \).

2) Mix algorithm:

\[ \text{Value}(i) = \left(n_i^{1}\right)/\left(t_i^{2} \cdot s_i^{3}\right) \]

The value of object \( i \) in the Mix algorithm [6] is calculated using formula 2, where \( n_i \) indicates the number of references after object \( i \) is stored in the cache. Frequent references increase value \( (i) \). In addition, \( t_i \) is the difference from the current time since object \( i \) was last referred, \( s_i \) is the size of the objects, and \( r1, r2, r3 \) are variables used to impose weight to each value and are set using a random constant. In the experiments described in this paper, \( r1, r2, r3 \) are given the same weight of “1” according to the SVC content. The time complexity of the Mix algorithm is \( O(n) \).

III. MOTIVATION

Because HTTP has been successful, HTTP Adaptive Streaming(HAS) using standard HTTP Web technology and reflecting the network status adaptively has received significant attention. An adaptive streaming media server of HAS divides and stores an encoded output at various bitrates. This segment is called “chunk.” While HAS is performed, a client requesting content determines the bitrate of the next chunk through a consideration of the bitrate of the currently requested chunk and the current network state. In addition, the HAS bitrate selector checks the buffer fullness of the clients, as well as the status of the network; for example, the volume of data stored in a full buffer constantly decreases, and judges whether the network is currently busy and lowers the bitrate of the requested video chunk. On the other hand, if buffer fullness is maintained over the threshold value, it judges whether the network status is currently free, and it can request a higher video chunk bitrate. When HAS performs adaptive streaming with no separated video data layer, the operating method of the CCN node is relatively simple. If the bitrate of the video chunk that the CCN node has cached has the same bitrate as the video chunk requested by a user, the CCN node transmits a requested data packet to the client that requests it. However, the bitrate is different, and contents transmitted from the content server or other CCN nodes.

This will cause an overcrowding transmission of the contents data. When layered video is used for adaptive video streaming through the use of H.264/SVC, the data from the base layer and several enhancement layers are formed into an independent data packet for transmission. When a request is made for content stored in a CCN node, and when the bitrate of the requested chunk is lower than the bitrate of an already stored chunk, a CCN node sends a base layer of the corresponding content and an enhancement layer up to the requested level. In contrast, when the bitrate of requested chunk is higher than the bitrate of the already stored chunk, it sends the data packet matched in the Content Store and requests an additionally required packet of the upper level of the enhancement layer from the content server or nearby CCN node. Then, each CCN cache hit rate decision made by cache replacement policy. Therefore, we adapt a new cache-replacement algorithm based on the CCNx [7], the reference implementation of CCN and
imposes the value in the proposed algorithm based on each object in consideration of the sequence, reference frequency, and trend. The cache reuse rate can be increased using a value based algorithm that deletes the data stored in the cache depending upon the value.

IV. LAYERED CACHE-REPLACEMENT ALGORITHM

When adaptive streaming is performed using H.264/SVC, the nearer the object is to the base layer, the higher the possibility of its reference compared with an object in an enhancement layer owing to the subordinate features of the layers. In addition, a rapid respond speed of the base layer is guaranteed, and a higher Quality of Experience (QoE) on average can be provided to the users. Considering this point, a new cache replacement policy was proposed. Furthermore, LGD-size, or Lmix algorithm is applied with a weighted value through the data packet of the enhancement level that each SVC video data object has based on this weighted value.

Algorithm 1 shows the basic operation sequence of the layered cache-replacement algorithm. For an SVC video packet, the value is determined based on the application of the weight by each layer. The weighted value of a layer multiplied by value(i) of each object can be adjusted by changing the value of “C”. When an object needs to be deleted, object i with the least value(i) is deleted. When object i with least value is searched, the value of all objects stored in the Content Store, or of a object, can be updated.

A. LGD-size Algorithm

\[
Value(i) = \frac{(l_m + C - l_i) \cdot n_i}{l_i \cdot (l_m + C)} \quad (3)
\]

The initial value of object i in the LGD-size algorithm is set as in formula 3. The Maxlayer is the value of the top bit rate enhancement layer determined during the encoding of the SVC content. The corresponding variable is set using the maximum value of the enhancement layer, and not the base layer. “C” indicates an error above “0”, which blocks Value(i) from being “0” and at the same time controls the weighted value of each layer when multiplying to value. L_i is the value of the enhancement layer of object i, and s_i is the size of object i. The initial value of the LGD-size algorithm imposes a higher value to the object of the lower layer. When objects are deleted in the same way as in the GD-size algorithm, Value(i) of the stored objects decreases by the same amount as their value, and when the objects are accessed again, the initial value is reset according to formula 3. The time complexity is O(log n). However, the operation method is changed when implemented, as shown in Figure 3, and the time complexity is O(1) during insertion/deletion of the object, and the time complexity of a regular value update is O(n).

B. Lmix algorithm

\[
Value(i) = \frac{((l_m + C - l_i) \cdot n_i)}{l_i \cdot (l_m + C)} \quad (4)
\]

The initial value of object i in the Lmix algorithm is set as in formula 4. The maxlayer is the value of the top bit rate of the enhancement layer, and “C” is the constant number with a value above “0”. The weighted value by the layer applied to Value(i) can be adjusted by changing the value of “C”, where l_i is the value of enhancement layer of object i, n_i is the number of references, t_i is the non-referred time, and s_i is the size of the objects. The Lmix algorithm calculates the value of each object when requiring a deletion of objects, and performs a cache replacement based on these values. The time complexity is O(n). Likewise, in the LGD-size algorithm, under the operation method shown in Figure 3, the time complexity is O(1) during the insertion/deletion of an object, and the time complexity of a regular value update is O(n).
V. IMPLEMENTATION

A. Implementation Environment

We implement the LFU, LRU, LGD-size, and Lmix algorithm for the CCNx-0.6.2 version, an open source of CCN and uses Linux Kernel version 2.6.32-279.2.1 of CentOS 6.3.

B. Cache Replacement Implementation

1) Cache replacement for CCNx: It is necessary to modify the code to reflect it in a CCN structure with the cache-replacement algorithm used in the existing CCNx. When objects are generated, deleted, or initialized by connecting objects stored in the Content Store, which is implemented inside the CCNx code, and the material structure is additionally implemented, the LFU, and LRU algorithms are implemented by referring to the additionally implemented material structure. This implementation method is applied to the LGD-size, and Lmix algorithms, as well as to the FIFO, LFU, and LRU algorithms. In addition, the GD-size and Mix algorithms, the latter of which is the reference of the proposed algorithm, are designed to apply a proxy server cache of a Content Delivery Network(CDN). A respectable algorithm is set as the fetch cost, which requires the cost when storing an object into cache, and reflects the value imposed on the objects. However, for data received from the CCN node, the data can be received from each CCN node, and not from the host, and the corresponding cost cannot be checked. Therefore, fetch cost is not considered in this paper as a cache-replacement algorithm applied to a CCN node.

2) Update scheme for object values: When each object to be stored in the Content Store of the CCN node uses a value based algorithm, the initial value is imposed depending on the algorithm used when stored in the cache. The value of the stored objects in the two proposed cache algorithms is updated when the stored objects are deleted or accessed. However, when adaptive video streaming is performed using H.264/SVC on a CCN, multiple interest packet and data packets are transmitted. Whenever objects stored in cache are accessed or deleted upon request, the updating of the values of every object stored in the Content Store is inefficient, and reduces the packet processing performance of each CCN node. Therefore, the value of the objects stored in the cache is not updated by every packet, but rather updated on a regular basis. In addition, the average value of the objects stored in the cache is calculated when the values are updated, and the value corresponding to a certain rate of the average value is set as a critical value.

The critical values are set to 20%, 40%, 60% and 80% of the average value. When implemented, a queue composed of five connection lists is generated and divided into five steps, where each step has an interval of 20%. Objects stored in the cache are moved to each connection list depending on its value. When objects stored in the cache need to be deleted, an object in the connection list with a value below 20% of the average value starts being deleted, and the next objects with the following least value are then deleted in sequence. Stored objects that are not deleted in the cache remain in a respectable queue until the next value update. Figure 3 shows a sequence of objects stored in each queue being deleted depending on their value. When objects inside the cache are to be deleted, objects with the lowest value or less than 20% of the average value are then deleted from the queue. If all objects with values of less than 20% of the average value are deleted and additional deletions are required, objects with values of less than 40% of the average values are then deleted from the queue in order. A value update of the new objects is applied using the LGD-size and Lmix algorithms. By modifying the operation method of the existing algorithm, a value update of an object stored in the cache is efficiently performed.

VI. EXPERIMENT

Under an environment in which the same content is repeatedly requested by the existing and proposed algorithms, the hit ratio and bitrate of the cache for each CCN node are measured.

A. Experiment Environment

The experiment environment is set up using one content server and three CCN nodes, as shown in Figure 4 and Dummynet [8] is used to limit the network bandwidth of the transmission. We aim to show that video streaming can be performed efficiently during a network bottleneck from a high volume of transmitted Internet traffic using a cache replacement policy. For this purpose, the bandwidth between the CCN_2 node and CCN_3 node is fixed at 0.3Mbps, which matches with bitrate of the base layer.

It is also assumed in this experiment that the cache size of each CCN node is only able to cache 40% of the overall data contents, and clients request the same content repeatedly and sequentially. Furthermore, the LGD-size and Lmix algorithms, which reflect the weighted value of objects stored in the cache at each layer, have a fixed C value of “3”, reflecting the weighted value.

1) Content server: The content server uses the OpenSuse. The video clip “Big Buck Bunny” [9] provided by Microsoft is encoded and stored using five steps for the SVC content, which has one base layer and four enhancement layers. The SVC data are separated by each layer, and the bitrate is 221-1503Kbps. For 221Kbps, the base layer contains the information, and for 429, 683, 1050, and 1503Kbps the enhancement layer. All data are divided and stored as a chunk, in a 2sec streaming unit and transmitted to a packet of a respectable layer upon request of the user (interest packet).

2) CCN node: OpenSUSE Linux OS is used, and the same cache replacement policy is selectable at each node. The CCN_2 node is fixed with a communication bandwidth of 0.3Mbps and a delay of 10ms with the CCN_3 node using Dummynet.
3) Client: Windows XP is used on the client devices. The requested bitrate depends adaptively on the current status of the network by means of the HAS bitrate selection algorithm [10].

B. Experiment Result and Analysis

This implementation method is applied to the LGD-size, and Lmix algorithms as well as the FIFO, LFU, and LRU algorithms. Experiments were conducted to check the cache hit ratio depending on the same adaptive video streaming, and to then compare each cache hit ratio when the existing FIFO, LFU, and LRU algorithms were used along with the LGD-size, and Lmix algorithms.

Figure 5(a) shows the cache hit ratio from each cache-replacement algorithm upon repetitive request at the CCN_1 node. For first video streaming, the cache hit ratio is “0%” since there is no value stored in the cache. In addition, for the FIFO algorithm, when repetitive requests are made for every algorithm, the cache hit ratio shows extreme results ranging from “0%” to “40%”. The reason for this result is that a low bitrate of the video packet is stored for the first request, and low bitrate is used for all hits for the second request, and the maximum hit ratio is therefore shown. However, the low bitrate of the cache in the first request is lowered according to a “first in first out” policy, and the minimum hit ratio is shown for the third request.

The LFU algorithm is advantageous under a scenario in which content is requested repetitively, but shows a low performance at a low cache size. As an example of this, in the experimental results using the LFU algorithm, the second and third video streaming show a high cache hit ratio, however, the fourth and fifth video streaming show a low cache hit ratio. From the second and third stages of the experiment, repetitively requested packets from the base layer and low enhancement layer are all stored in the cache and show high cache efficiency. However, when the cache size is small, objects are frequently deleted, and thus the LFU algorithm shows its limitation when the base layer or low enhancement layer is deleted. The reason for this result is that even when a packet from a respectable layer is received again after a deletion, it is not restored, and can be deleted from the cache easily since it has a low reference frequency in terms of its competition with other packets. Therefore once an object is deleted from the cache, it is difficult to store in the cache again.

Of the existing algorithms, the LRU algorithm shows the highest cache hit ratio on average. However, a high bitrate of the content is requested only once, and a base layer packet with a high rate of repetitive requests can be deleted. Based on this factor, the fourth video streaming shows a low cache hit ratio. This is more likely to occur regularly when a repetitive request is performed. Similar to the LRU algorithm, the LGD-size algorithm reflects reference the reference trend in value and deletes the object stored in the cache. When the weighted value for each layer is reflected in the objects stored in the cache, the results show that the overall cache hit ratio is higher than in the LRU or LFU algorithm. In addition, the proposed cache-replacement algorithm used in this experiment showed the lowest cache hit ratio in terms of deviation at different times of the experiment. Lastly, the Lmix algorithm, when considering the reference frequency, latest reference trend, size of the objects and layer, showed the highest cache hit ratio. To summarize, the LGD-size algorithm and Lmix algorithms showed a 5 to 15% higher cache hit ratio than the existing cache-replacement algorithm. In particular, this tendency is maintained even when video streaming is repeated. In addition,
since CCN\_1, which is the nearest node, has a high cache hit ratio, the benefit of reducing the amount of network traffic is expected.

Figure 5(b) shows the cache hit ratio for each cache-replacement algorithm depending on the repetitive requests at the CCN\_2 node. For the FIFO algorithm, a “0%” cache hit ratio is shown in almost all cases. The reasons for this are that the same data packet is cached at the same time as a chunk with CCN\_1. By requesting a data packet that is not stored in CCN\_1 at the time when the LFU algorithm at CCN\_2 is requested, the general reference frequency of the stored objects at CCN\_2 is lower, therefore resulting in a low cache hit ratio. The LRU algorithm showed an average cache hit ratio of approximately 15% with no differences in the number of requests. The reason for this is that an overlapped data packet with CCN\_1 is reduced by reflecting the latest reference trend. This result can also be seen in the experimental results of LGD-size algorithm. When adaptive video streaming is performed, and when a once requested contents object is the objects of the subordinate SVC layer, a repetitive call is likely to be transmitted, which in turn incurs a higher cache hit ratio. The LGD-size and Lmix algorithms first delete the higher bitrate enhancement layer packet that has a low possibility to be requested according to the value reflected with the weighted value between the layers, and then store the lower bitrate enhancement layer packet that has a high possibility to have been requested again, therefore showing a higher cache hit ratio.

In particular, the two proposed algorithms show a consistently increasing trend depending on the order of the request, which shows that an overlap of the cache with a nearby CCN node is reduced.

Figure 5(c) shows the cache hit ratio for each cache-replacement algorithm depending on repetitive requests at the CCN\_3 node. The CCN\_3 node is the nearest node to the contents server and shows an average low cache hit ratio of 5% for almost all cases except the Lmix algorithm, which shows a 10% cache hit ratio, which is a relatively higher rate of repetitive requests. The reasons for this is that the reference frequency, latest reference trend, and SVC layer are reflected, resulting in a constant increase according to the number of trials. From the results shown in the graphs in Figure 5(a), and (b), as well as in Figure 5(c), node(CC\_N\_1) near a user shows high cache efficiency, whereas node(CC\_N\_3) near the content server shows relatively low cache efficiency.

The graph in Figure 5(d) shows the cache hit ratio for the number of interest packet that occur at the client by cache-replacement algorithm, and for the combined number of cache hits that occur at all CCN\_1-3 node. Likewise, for the hit ratio at each node, the FIFO algorithm shows a trend of repetitively switching from higher and lower cache hit ratios. Its shortages lies in the fact that objects previously stored in the cache are difficult to be utilized at a later stage. For the second video streaming, all cache replacement show a similar cache hit ratio. The reason for this is that the same cache size and cache replacement do not occur. For the fourth video streaming, the LFU and LRU algorithms show a low cache hit ratio since the low layer packet was deleted during the third video streaming owing to the features of each algorithm. On the other hand, the LGD-size and Lmix algorithms consider the reference frequency, latest reference trend, and layer value of the SVC contents objects, and a low layer packet is not deleted in the cache, which consistently increases the cache hit ratio.

VII. Conclusion

We proposed cache replacement algorithms to enhance the efficiency of H.264/SVC based video streaming in a CCN test-bed. To compare the proposed algorithms(LGD-size, and Lmix) three other algorithms(FIFO, LFU, and LRU) were applied in a CCN. For the LGD-size algorithm used in the proposed cache replacement algorithm, the layer of the SVC content is reflected in the GD-size algorithm, which is a value based cache replacement policy reflecting the size of an object stored in the cache and the latest reference trend, and the value is then determined and the cache is replaced according to the respectabe value. For the Lmix algorithm, cache replacement is performed using the value reflected layer of the SVC content in the Mix algorithm, which reflects the size of the objects stored in the cache and the accumulated number of latest references. The experiment results showed that the proposed LGD-size and Lmix algorithms have a maximum of double the cache hit ratio for each CCN node compared with the three existing algorithms (FIFO, LFU, LRU). The results show that the proposed LGD-size and Lmix algorithm can provide fast and stable video streaming service, and transmissions can be conducted with better quality when the bitrate is selected. In this way, the layered cache replacement algorithm can contribute to an improved performance of adaptive video streaming. For further research, an analysis of the efficiency of the cache replacement algorithm on diverse network topologies is required.

ACKNOWLEDGMENT

This work was in part supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No.2010-0029180) with KREONET and in part supported by Seoul R&BD Program (WR080951).

REFERENCES