Received February 27, 2019, accepted March 25, 2019, date of publication April 9, 2019, date of current version April 18, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2909563

Progressive Caching System for Video Streaming Services Over Content Centric Network

HYUNMIN NOH AND HWANGJUN SONG

Department of Computer Science and Engineering, Pohang University of Science and Technology (POSTECH), Pohang-si 790-784, South Korea

Corresponding author: Hwangjun Song (hwangjun@postech.ac.kr)

This work was partly supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. 2015-0-00567, Development of Access Technology Agnostic Next-Generation Networking Technology for Wired-Wireless Converged Networks) and National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2016R1A2B4007812).

ABSTRACT This paper presents a metafile-based progressive caching system over the content-centric networking (CCN) tree that supports seamless video streaming services with a high network utilization. In the proposed caching system, each CCN node uses a metafile made by a scalable caching algorithm for efficient and fast chunk caching management, and the reserved area of the CCN interest/data packet headers is used to deliver caching information among the CCN nodes. Based on this caching information, the proposed caching system determines the caching range of video data to minimize the required peak bandwidth for each link. The proposed caching system is implemented using the NS-3 based named data networking simulator. Furthermore, a real cellular wireless network testbed is realized with C/C++, open sources such as CCNx and Ubuntu MATE, and a Raspberry PIs to examine the performance of the proposed caching system. The experiment results demonstrate the performance improvement achieved by the proposed caching system.

INDEX TERMS Video caching, channel bandwidth, content centric networking, video streaming.

I. INTRODUCTION

In recent years, the demand for video streaming services over wired/wireless networks has been increasing rapidly. According to the Cisco visual network index [1], overall data traffic is expected to grow at a compound annual growth rate (CAGR) of 24% from 2016 to 2021, reaching 3.3 ZB per year, or 278 EB per month, which represents nearly a threefold increase compared to 2016. The main catalyst for this exponential rise in data traffic is the explosive increase in over-the-top (OTT) media services such as YouTube, Netflix, and Hulu. In [1], Cisco estimates that video traffic will increase at a CAGR of 25% from 2016 to 2021 and account for 82% of total data traffic in 2021. Unfortunately, this explosive growth in video traffic may cause a traffic overload problem on the limited capacity of wired/wireless networks, which degrade the user-perceived video quality. It is still quite challenging to support this explosive data traffic growth in wired/wireless networks. To solve this problem, various approaches have been considered to alleviate network traffic [2]–[6], e.g. web cache [7], content delivery networks (CDNs) [8], and Peer-to-Peer (P2P) [9] networks, etc. However, their performance is limited due to their operations at the application layer and the deployments that do not share a common naming or encoding of their data units [10], [11].

The Content Centric Networking (CCN) [12] technology is an emerging technology to efficiently handle rapid traffic growth by accessing content with its name instead of the physical location of data, and realizing the in-network caching. In a CCN, content is divided into chunks (i.e., CCN data packets) and each chunk has its own identified name. A content consumer requests content through a CCN interest packet, and any node with the requested content immediately provides the content to the consumer through a CCN data packet. A CCN node includes a packet forwarding engine consisting of a Forwarding Information Base (FIB), a Pending Interest Table (PIT), and a Content Store (CS). The FIB stores a face list that interest packets should be forwarded. The PIT saves the face at which the CCN interest packet arrives, and the requested CCN data packets are forwarded to the corresponding face. The CS is a cache space to store arriving CCN data packets. So far, considerable amounts of research efforts have been devoted to alleviating the network traffic in the CCN architecture. In [13], Psaras et al. proposed a probabilistic in-network caching scheme. It approximates the capability of paths based on their lengths, and

The associate editor coordinating the review of this manuscript and approving it for publication was Byung-Seo Kim.

2169-3536 © 2019 IEEE. Translations and content mining are permitted for academic research only.
Personal use is also permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

caches contents probabilistically to reduce caching redundancy. In [14], Bernardini et al. introduced a simple, but effective caching algorithm that caches only popular contents in order to achieve a higher cache hit ratio and reduce the cache load at each node. In [15], Cho et al. proposed a collaborative caching algorithm that considers content popularity. In this algorithm, an upstream node recommends the number of chunks to be cached at its downstream node by reflecting the content popularity. In [16], Suksomboon et al. proposed a PopCache that caches contents more or less in accordance with a popularity characteristic. In [17], Jing et al. proposed a distributed max-gain in-network caching strategy for information-centric networks. They focused on maximizing local cache gain and minimizing the cache replacement penalty to improve the cache hit ratio. In [18], Wang et al. proposed a cache allocation algorithm that distributes cache capacity across routers under a constrained total storage budget.

Until now, many effective caching and replacing algorithms for video streaming services have been studied. In [19], Sung et al. proposed an efficient cache placement strategy in a two-tier wireless content delivery network; in such a scheme, the cache servers are deployed in small cell mesh networks as well as a data center to overcome the overload problem. Li et al. [20] introduced a caching scheme for adaptive video streaming to reduce total video distortion by considering the rate-distortion characteristics of multiple bitrate videos and the coordination among distributed cache servers. In [21], Su et al. proposed a game theory-based layered video caching scheme for multiple social groups with limited caching capacity to achieve a higher cache hit ratio and user QoE of each social group. Miao et al. proposed a scalable selective caching system for QoS networks (SCQ) [22]. The SCQ algorithm determines a caching frame sequence to prevent client buffer underflow with low network bandwidth cost. In [23], Oh et al. proposed a metafile based scalable caching system, which is designed to minimize the required peak bandwidth depending on relative caching frame positions as well as frame sizes. However, when these algorithms are employed on multiple caching nodes, the unnecessary or redundant caching may increase because they consider only single caching node such as a proxy server.

In this paper, we propose a metafile based progressive caching system for seamless video streaming services over the CCN tree. The major contributions of this manuscript are summarized in the following:

- A metafile created by a scalable caching algorithm [23] is employed in the CCN architecture. The metafile is designed to reduce the required peak bandwidth for seamless streaming services according to the cached chunks along the content delivery route.
- An efficient caching information exchanging protocol with no additional network overhead is designed among CCN nodes by using the reserved header area of the CCN interest/data packets.

TABLE 1. An example of the metafile for scalable caching over CCN tree.

Priority	Chunk Index	Chunk Size	Required Bandwidth
-	0 (no caching)	-	96204.37
1	1	8000	96032.27
2	2	8000	95860.17
3	3	8000	95688.07
4	559	8000	95515.97
5	546	8000	95343.87
6	545	8000	95171.76
788	234	8000	100.34
789	56	8000	0

- A progressive caching system effectively determines the caching range based on the above metafile for the seamless video streaming services. Chunks of video are selectively stored at each node by a progressive caching module.
- For performance verification in large-scale CCN environments, we use the NS-3 based named data networking simulator (ndnSIM). In addition, for performance verification under a real cellular wireless environment, the proposed caching system is fully implemented using CCNx and C/C++ on a single-board computer, called Raspberry Pi.

The rest of this paper is organized as follows. The details of the proposed progressive caching system over the CCN tree are presented in section II, and experimental results are given in section III. Finally, the concluding remarks are provided in section IV.

II. PROPOSED PROGRESSIVE CACHING SYSTEM OVER CCN TREE

The goal of the proposed caching system is to provide seamless video streaming services with high network utilization over the CCN tree. The overall architecture of the proposed caching system is illustrated in Fig. 1. As shown in the figure, the CCN node includes three components: the CCN forwarding engine, the caching information exchanging module, and the progressive caching module. The CCN forwarding engine

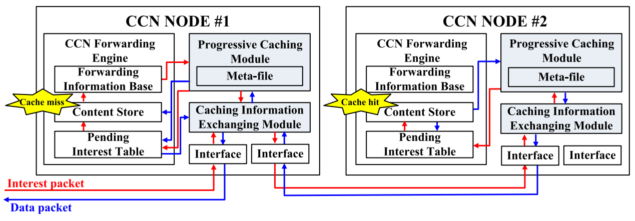


FIGURE 1. Architecture of the proposed caching system (red and blue lines indicate the packet flow when CCN node #2 caches the requested data).

performs basic CCN operations as mentioned earlier. The caching information exchanging module updates the caching status of the CCN nodes through the interest/data packet headers, and the progressive caching module manages cached chunks in the CS using metafiles and the caching information received from other CCN nodes. There are two main differences between the proposed caching system and the original CCN. First, the proposed caching system exchanges caching information among CCN nodes through the modified header structure of interest packet and data packet, whereas the original CCN does not share any information among CCN nodes. Second, the proposed caching system is implemented to support a selective caching scheme in order to reduce unnecessary redundant caching at neighboring nodes by using metafiles and exchanging caching information. The metafiles provide caching priority to minimize the required peak bandwidth and alleviate redundant caching over distributed CCN nodes. In the followings, the metafile structure for CCN is discussed in section 2.A, a problem description is provided in section 2.B, and the details of the proposed caching system over the CCN tree are presented in section 2.C.

A. METAFILE STRUCTURE FOR CCN

In this section, we describe how to create the metafile for the CCN environment based on scalable caching algorithm [23]. It is assumed that video frames are compressed in variable sizes and transmitted in chunk format over the CCN tree. Therefore, a chunk may contain multiple frames, or a compressed frame may be split into multiple chunks. Since a video frame can only be displayed after all the chunks containing the frame data arrive, the required peak bandwidth should be calculated in the chunk unit, which is stored in the metafile. At first, some symbolic descriptions are provided to the metafile structure for the CCN. The total size of the video #m to provide seamless streaming service at the time t is calculated by

$$s_m^{frame}(t) = \sum_{k=1}^{\lfloor t/T^{frame}+1 \rfloor} R_{m,k}, \tag{1}$$

where $R_{m,k}$ is the size of the k_{th} frame for video #m, and the time interval between two adjacent frames is T_m^{frame}. The number of chunks for the seamless streaming service at the time t is represented by

$$n_m^{chunk}(t) = \left\lceil \frac{s_m^{frame}(t)}{S^{chunk}} \right\rceil, \tag{2}$$

where S^{chunk} is the size of the chunk. When the video #m consists of N_m^{chunk} chunks, the cached chunks along the interest packet forwarding route are represented by

$$\vec{c}_m = \left(c_{m,1}, \ldots, c_{m,k}, \ldots, c_{m,N_m^{chunk}}\right), \tag{3}$$

$$c_{m,k} = \begin{cases} 1, & \text{if the } k_{th} \text{ chunk of the video \#}m \text{ is cached,} \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

The number of chunks downloaded through the content server-side link due to cache misses is calculated by

$$n_{\vec{c}_m}^{down}(t) = \sum_{k=1}^{n_m^{chunk}(t)} \left(1 - c_{m,k}\right). \tag{5}$$

The required peak bandwidth on the content server-side link for the seamless streaming service for video #m is defined by

$$BW_m^{link}(\vec{c}_m) = \max_{1 \le t \le T_m^{frame} \cdot N_m^{frame}} \left\{ \frac{n_{\vec{c}_m}^{down}(t) \cdot S^{chunk}}{t} \right\}, \tag{6}$$

where N_m^{frame} indicates the number of frames in video #m. The time that the peak bandwidth is required for video #m is obtained by

$$t_m^{peak}(\vec{c}_m) = \arg\max_{1 \le t \le T_m^{frame} \cdot N_m^{frame}} \left\{ \frac{n_{\vec{c}_m}^{down}(t) \cdot S^{chunk}}{t} \right\}. \tag{7}$$

The maximum buffer size to prevent buffer overflow for video #m at the client is calculated by

$$B_{ccn}(\vec{c}_m) = \max_{1 \le t \le T_m^{frame} \cdot N_m^{frame}} \left\{ t \cdot BW^{link}(\vec{c}_m) - n_{\vec{c}_m}^{down}(t) \cdot S^{chunk} \right\}, \tag{8}$$

and the normalized buffer size is defined by the decreased maximum buffer size after caching a chunk divided by its chunk size as follows.

$$B_{ccn}^{norm}(\vec{c}_m, k) = \frac{\left(B_{ccn}(\vec{c}_m)|_{c_{m,k}=0} - B_{ccn}(\vec{c}_m)|_{c_{m,k}=1}\right)}{S^{chunk}}, \tag{9}$$

where $B_{ccn}(\vec{c}_m)|_{c_{m,k}=0}$ and $B_{ccn}(\vec{c}_m)|_{c_{m,k}=1}$ are the maximum buffer size before and after caching the k_{th} chunk of video #m, respectively. Now, we make the metafile for the CCN environment as follows.

Step 1) Find the k_{th} chunk maximizing $B_{ccn}^{norm}(\vec{c}_m, k)$ for $1 \le t \le t_m^{peak}$. Then, store the caching priority, chunk number, chunk size, and required peak bandwidth after caching the chunk in the metafile.

Step 2) When several chunks have the same normalized buffer size, find a chunk minimizing the normalized buffer size $B_{ccn}(\vec{c}_m)$. Then, store the caching priority, chunk number, chunk size, and required peak bandwidth in the metafile.

The caching priority, chunk number, chunk size and required peak bandwidth determined through the above steps are stored in the metafile as shown in Table 1. In the proposed caching system, the metafiles are generated by the service provider, and distributed to all CCN nodes.

B. OPTIMAL PROBLEM DESCRIPTION

Now, we formulate the caching range optimization problem to support seamless video streaming services with high network utilization over the CCN tree. Most existing caching algorithms [22], [23] store chunks of content in order of their priority at a single proxy node. In contrast, we assume distributed caches in the tree structure over the CCN. In this case, if the CCN relay nodes cache contents independently, the existing algorithms may cause inefficient duplicate chunks caching among CCN nodes. Therefore, the problem is formulated to efficiently determine the caching range at each CCN node while considering dependencies among distributed caches to avoid unnecessary duplicate chunks. Before presenting a detailed description of the proposed caching system, some symbolic descriptions are given in the following.

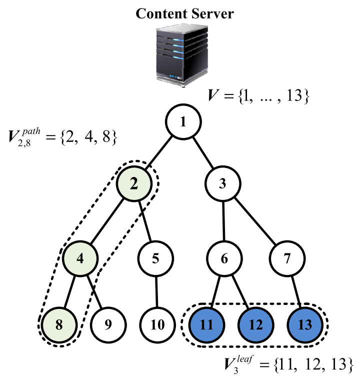


FIGURE 2. An example of the CCN tree topology under consideration.

The CCN tree topology under our consideration is illustrated in Fig. 2. We assume that a root node is the content server, and all the interest packets are forwarded from each leaf node to the root node. $\boldsymbol{V}$ is the set of all CCN nodes over the CCN tree, $\boldsymbol{V}_i^{leaf}$ is the set of all leaf nodes associated with the node i, and $\boldsymbol{V}_{i,j}^{path}$ is the set of all nodes along the interest packet forwarding route between node i and node j. The caching range of a CCN node is represented by $\left(l_m^i, h_m^i\right)$, where l_m^i and h_m^i indicate the lowest and the highest priority values of the cached chunks for the video #m at the node i, respectively. Now, the following condition should be satisfied among node i and its descendant nodes to achieve consecutive caching in the metafile without missing parts.

$$h_m^i \leq \max_{x \in \boldsymbol{V}_{i,j}^{path} - \{i\}} \left\{ l_m^x \right\} \quad for \forall m \in \boldsymbol{M},\ \forall i \in \boldsymbol{V},\ \forall j \in \boldsymbol{V}_i^{leaf}. \tag{10}$$

And, the total amount of peak bandwidth required for the seamless display of the video #m over a link between node i and its parent node with no cache is calculated by

$$\sum_{j \in \mathbf{V}_i^{leaf}} \lambda_m^j \cdot bw_m^{sc} \left(\max_{x \in \mathbf{V}_{i,j}^{path}} \left\{ l_m^x \right\} \right), \tag{11}$$

where $bw_m^{sc}\left(\max_{x \in \boldsymbol{V}_{i,j}^{path}} \left\{ l_m^x \right\} \right)$ is the required peak bandwidth corresponding to the priority value $\max_{x \in \boldsymbol{V}_{i,j}^{path}} \left\{ l_m^x \right\}$ in the metafile, λ_m^j is the request rate for the video #m from the leaf node j. Fig. 3 illustrates a tree example for the calculation of the amount of required peak bandwidth at a link.

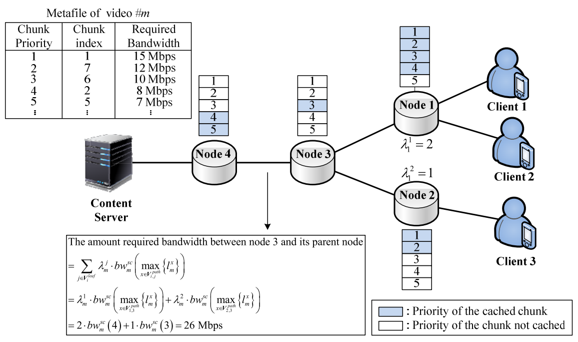

Metafile of video #m
Chunk Priority | Chunk index | Required Bandwidth
1 | 1 | 15 Mbps
2 | 7 | 12 Mbps
3 | 6 | 10 Mbps
4 | 2 | 8 Mbps
5 | 5 | 7 Mbps
⋮ | ⋮ | ⋮
Content Server
Node 4
Node 3
Node 1
Node 2
Client 1
Client 2
Client 3
$\lambda_1^1 = 2$
$\lambda_1^2 = 1$
1 2 3 4 5
The amount required bandwidth between node 3 and its parent node
$= \sum_{j \in V_i^{leaf}} \lambda_m^j \cdot bw_m^{sc}\left(\max_{x \in V_{i,j}^{path}}\{l_m^x\}\right)$
$= \lambda_m^1 \cdot bw_m^{sc}\left(\max_{x \in V_{1,3}^{path}}\{l_m^x\}\right) + \lambda_m^2 \cdot bw_m^{sc}\left(\max_{x \in V_{2,3}^{path}}\{l_m^x\}\right)$
$= 2 \cdot bw_m^{sc}(4) + 1 \cdot bw_m^{sc}(3) = 26$ Mbps
: Priority of the cached chunk
: Priority of the chunk not cached

FIGURE 3. An example to calculate the amount of required peak bandwidth.

Now, we can formulate the optimal problem to achieve our goal as follows.

Optimal Problem Formulation: Determine h_m^i and l_m^i for $\forall i \in \boldsymbol{V}, \forall m \in \boldsymbol{M}$ with the tree topology information to minimize the total amount of required peak bandwidth

$$\sum_{m \epsilon \boldsymbol{M}} \sum_{i \epsilon \boldsymbol{V}} \sum_{j \epsilon \boldsymbol{V}_i^{leaf}} \lambda_m^j . bw_m^{sc} 0 \left(\max_{x \epsilon \boldsymbol{V}_{i,j}^{path}} \left\{ l_m^x \right\} \right), \tag{12}$$

$$subject\ to \sum_{m \in \boldsymbol{M}} \left(l_m^i - h_m^i \right) \leq S_i^{cs} \quad for\ \forall i \in \boldsymbol{V}, \tag{13}$$

$$and\ h_m^i \leq \max_{x \in \boldsymbol{V}_{i,j}^{path} - \{i\}} \left\{ l_m^x \right\} \quad for\ \forall m \in \boldsymbol{M},$$

$$\forall i \in \boldsymbol{V},\ \forall j \in \boldsymbol{V}_i^{leaf}, \tag{14}$$

where S_i^{cs} is the CS capacity of the node i and $\boldsymbol{M}$ is the set of caching videos over the CCN tree. However, it is very difficult to obtain the optimal solution because the whole tree topology information and the caching status of all CCN nodes are needed. Furthermore, the computational complexity and the amount of control overhead increase exponentially according to the number of videos and nodes. The computational complexity of the full search for the optimal solution is $O\left(n^{2|\boldsymbol{V}||\boldsymbol{M}|}\right)$, and the amount of control overhead to deliver the global caching information is $2\,|\boldsymbol{V}|\,|\boldsymbol{M}|$.

C. PROPOSED PROGRESSIVE CACHING SYSTEM

We simplify the above optimal problem to acquire a near optimal solution in a distributed way with a low computational complexity and no additional control overhead. In the proposed caching system, each CCN node determines its own caching range in a distributive way, and effectively shares caching information with other nodes. The proposed system has no additional control overhead since it employs the reserved field of CCN interest/data packet headers to exchange caching information among CCN nodes. When new caching information arrives, each node updates its own caching range. These processes work iteratively to converge a near optimal solution. Compared with the optimal solution of

Protocol 1 When an Interest Packet Arrives at the Node i

```
Node $i$ receives an interest packet for the chunk with priority $p$ in the video #$m$ with $l^{interest}$
If $l_m^i > l^{interest}$
    $l^{interest} \leftarrow l_k^i$
End IF
If $l_m^{interest} > l^{interest}$
    $l_m^{interest} \leftarrow l^{interest}$
End IF
If Cache miss
    Forward the interest packet with $l^{interest}$ to the next node
    $\lambda_{j,k}^i \leftarrow \lambda_{j,k}^i + 1$
End IF
```

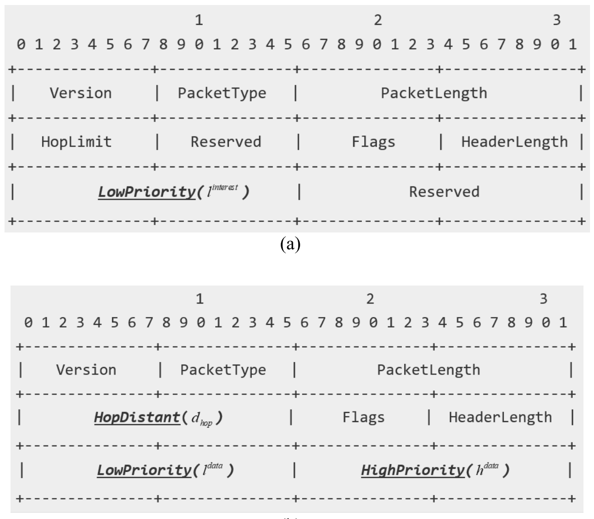
Protocol 2 When a Data Packet Arrives at the Node i

```
Node $i$ receives a data packet for the chunk with priority $p$ in the video #$m$ with $l^{data}$, $h^{data}$, $d^{hop}$
If $h^{data} \le l_m^i \le l^{data}$
    $h^{data} \leftarrow h_m^i$
End IF
If $h_m^{data} > h^{data}$
    $h_m^{data} \leftarrow h^{data}$
End IF
$d_{m,p}^{hop} \leftarrow d^{hop}$
$d^{hop} \leftarrow d^{hop} + 1$
Forward the data packet with $l^{data}$, $h^{data}$, $d^{hop}$ to the next node
```

the above problem, we can avoid a disastrous increase of the computational load and the control overhead. In this section, we describe the proposed caching system over the CCN tree in detail. First, the caching information exchanging protocol, and a simplified problem formulation is given to obtain a near optimal solution in a distributed way. Finally, the caching range determining processes are described in detail.

1) CACHING INFORMATION EXCHANGING PROTOCOL

In the proposed caching system, the total amount of required peak bandwidth is associated with the lowest/highest priority values, hop distance, and request rate. To share caching information with CCN nodes, we propose a caching information exchanging protocol by modifying the reserved fields of the existing CCN interest/data packet headers. Fig. 4 represents the modified header structure of CCN packets. If the lowest priority value of the cached chunks at a node is lower than the ***LowPriority*** field ($l^{interest}$) of the arriving interest packet, then $l^{interest}$ is updated with the lowest priority of the cached chunks at the node. And the node store the minimum $l^{interest}$ for each video #m in $l_m^{interest}$. If the highest priority of the cached chunks at a node is between the ***LowPriority*** field (l^{data}) and the ***HighPriority*** field (h^{data}) of the arriving data packet, h^{data} is replaced with the highest priority of the cached chunks at the node. The ***HopDistance*** field (d_{hop}) is updated to $d_{hop} + 1$. And the node stores the minimum h^{data} for each video #m in h_m^{data}. By using these protocols, every node



```
                    1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
|    Version    |  PacketType   |         PacketLength          |
+---------------+---------------+---------------+---------------+
|   HopLimit    |   Reserved    |     Flags     | HeaderLength  |
+---------------+---------------+---------------+---------------+
|   LowPriority( l^interest )   |           Reserved            |
+---------------+---------------+---------------+---------------+
```

(a)

```
                    1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
|    Version    |  PacketType   |         PacketLength          |
+---------------+---------------+---------------+---------------+
|      HopDistant( d_hop )      |     Flags     | HeaderLength  |
+---------------+---------------+---------------+---------------+
|     LowPriority( l^data )     |    HighPriority( h^data )     |
+---------------+---------------+---------------+---------------+
```

(b)

FIGURE 4. **Modified header structure of CCN packets: (a) interest packet and (b) data packet (the changing parts are written in Italic font).**

Algorithm Caching Range Determining Process

```
Input: $l_m^{interest}$, $h_m^{data}$, $\lambda_{m,p}^i$, $d_{m,p}^i$ for $\forall m \in \boldsymbol{M}$ and $1 \le p \le N_m^{chunk}$
Output: $l_m^i$, $h_m^i$ for $\forall m \in \boldsymbol{M}$
For $\forall m \in \boldsymbol{M}$
    $h_m^i \leftarrow l_m^{\min}$, $l_m^i \leftarrow h_m^{\min}$
End For
For $\forall m \in \boldsymbol{M}$
    For $l_m^i + 1 \le x \le N_m^{chunk}$
        $\mu_m^i(x) \leftarrow \frac{\sum_{p=l_m^i}^{x} (bw_m^{sc}(p+1) - bw_m^{sc}(p)) \cdot \lambda_{m,p}^i \cdot d_{m,p}^i}{x - l_m^i}$
    End For
End For
$n_i^{cache} \leftarrow \sum_{m \in M} (l_m^i - h_m^i)$
$A$ = Sort $(m, x)$ in the descending order by $\mu_m^i(x)$
For all $(m, x) \in A$
    If $x > l_m^i$ and $n_i^{cache} + x - l_m^i \le S_i^{CS}$
        $n_i^{cache} \leftarrow n_i^{cache} + x - l_m^i$
        $l_m^i \leftarrow x$
    End IF
End For
Requests and caches the chunks within the caching range between $h_m^i$ and $l_m^i$ for $\forall m \in \boldsymbol{M}$
```

can obtain the hop distance, request rate of each chunk, and caching information. It is summarized in the following:

Fig. 5 shows how the proposed system delivers the lowest/highest priority values, hop distance, and the request rate of each chunk to nodes by using the modified CCN packets.

2) SIMPLIFIED PROBLEM DESCRIPTION

We simplify the above optimal problem to obtain a near optimal solution in a distributed way. When caching information

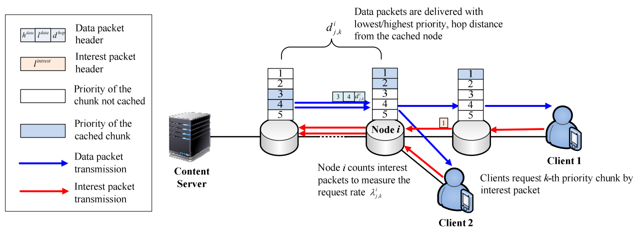

h^{data} l^{data} d^{hop}
Data packet header
$l^{intrest}$
Interest packet header
Priority of the chunk not cached
Priority of the cached chunk
Data packet transmission
Interest packet transmission
Content Server
$d^i_{j,k}$
Data packets are delivered with lowest/highest priority, hop distance from the cached node
1
2
3
4
5
3 4 $d^i_{j,k}$
Node i
1
Node i counts interest packets to measure the request rate $\lambda^i_{j,k}$
Client 1
Client 2
Clients request k-th priority chunk by interest packet

FIGURE 5. An example of interest and data packet delivery mechanism to provide the lowest/highest priority values, the hop distance, and the request rate to the node i.

such as the lowest/highest priority values, hop distance and request rate of each chunk is delivered at the node i as shown in Fig. 5, we can calculate the expected amount of reduced peak bandwidth by caching a chunk with priority p in the video #m at the node i as follows.

$$\left(bw^{sc}_m(p+1) - bw^{sc}_m(p)\right) \cdot \lambda^i_{m,p} \cdot d^{hop}_{m,p}, \tag{15}$$

where $\lambda^i_{m,p}$ is the request rate of the chunk with priority p in the video #m. $d^i_{m,p}$ is the hop distance between the cached node to the node i. Now, we simplify the above optimal problem as follows.

Simplified Problem Formulation at the Node i: Determine h^i_m and l^i_m for $\forall m \in \boldsymbol{M}$ to maximize the total amount of reduced peak bandwidth

$$\sum_{m\in\boldsymbol{M}} \sum_{p=h^i_m}^{l^i_m} \left(bw^{sc}_m(p+1) - bw^{sc}_m(p)\right) \cdot \lambda^i_{m,p} \cdot d^{hop}_{m,p}, \tag{16}$$

$$\textit{subject to} \sum_{m\in\boldsymbol{M}} \left(l^i_m - h^i_m\right) \le S^{cs}_i, \tag{17}$$

$$\textit{and } h^i_m \le l^{interest}_m \text{ and } l^i_m \ge h^{data}_m \quad \textit{for } \forall m \in \boldsymbol{M}. \tag{18}$$

3) CACHING RANGE DETERMINING PROCESS

We describe how to obtain a feasible solution of the simplified problem in detail based on the metafiles and the caching information exchanged among CCN nodes. In the proposed caching system, h^i_m and l^i_m are initialized by caching information delivered through interest/data packets. And then, l^i_m increases greedily to maximize the total amount of reduced peak bandwidth. The caching range determining process is performed iteratively from external nodes to the root node until the amount of required peak bandwidth converges. The details of the caching range determining process at the node i are presented below.

STEP 1) Initialization:

Initialize the highest/lowest caching priorities for $\forall m \in \boldsymbol{M}$. h^i_m and l^i_m are initialized to $l^{interest}_m$ and h^{data}_m, respectively.

STEP 2) Calculation of the average amount of reduced peak bandwidth per cacheable chunks:

Generate a possible set of (m, x) for $\forall m \in \boldsymbol{M}$ and $l^i_m + 1 \le x \le N^{chunk}_j$. For all generated sets of (m, x), calculate the average amount of reduced peak bandwidth per cacheable chunk $\mu^i_m(x)$ as follows

$$\mu^i_m(x) = \frac{\sum_{p=l^i_m}^{x} \left(bw^{sc}_m(p+1) - bw^{sc}_m(p)\right) \cdot \lambda^i_{m,p} \cdot d^i_{m,p}}{x - l^i_k}. \tag{19}$$

STEP 3) Determination of the caching range:

Find the maximum x for each video #m that satisfies the limited CS capacity constraint of Eq. (17) by examining all in descending order by $\mu^i_m(x)$. And then, l^i_m is set to x for each video #m.

III. EXPERIMENTAL RESULTS

During the experiments, the proposed caching system is implemented using the ndnSIM [26] for large-scale network environments and a real cellular wireless network testbed is realized with a Raspberry Pi [28] and CCNx [30]. We use three video sequences, "Elephants-Dream", "Of-Forests-And-Man", and "RedBull-Playstreet", with an average bit rate of 1Mbps and 24 frames per second. It is assumed that the video requests follow the Zipf distribution or uniform distribution. Zipf parameter is set to 1.1. The chunk size (S^{chunk}) is set to 8 KB. The total amount of required peak bandwidth is measured every 600 s and the caching range determining process is performed iteratively every 600 s. The maximum number of cacheable chunks (S^{cs}_i) is set to 700 at all CCN relay nodes.

A. PERFORMANCE COMPARISON WITH EXISTING CACHING SYSTEMS

In this section, we demonstrate the performance of the proposed caching system. In the simulations, we employ two k-ary tree topologies for large-scale environments as shown in Fig. 6. In literature, a k-ary tree has been used as a CCN tree topology to evaluate caching system performance [14], [15], [17], [24]. The asymmetric tree topology is composed of a content server, 23 CCN relay nodes, and 39 clients. The symmetric tree topology illustrated in Fig. 6 (b) is composed of a content server, 25 CCN relay nodes, and 45 clients. The available bandwidth for each link is set to 100Mbps and the transmission delay is set to 10ms.

Table 2 represents the total amount of required peak bandwidth for all content delivery routes. The route number indicates the route according to the order of depth-first. For example, Route #1 includes the CCN node #1 and #3, and Route #13 includes the CCN Node #2, #5, #6, and #12. As shown in Table 2, the total amount of required peak bandwidth decreases monotonically. This is because the proposed caching system always determines the caching range to minimize the total amount of required peak bandwidth.

Next, we compare the performance of the proposed caching system with that of five existing caching systems such as ProbCache [13], MPC [14], LRU [25], LFU [26], and the optimal solution obtained by CPLEX [34], which is a well-known software program for solving optimization problems. However, it is observed during the experiment

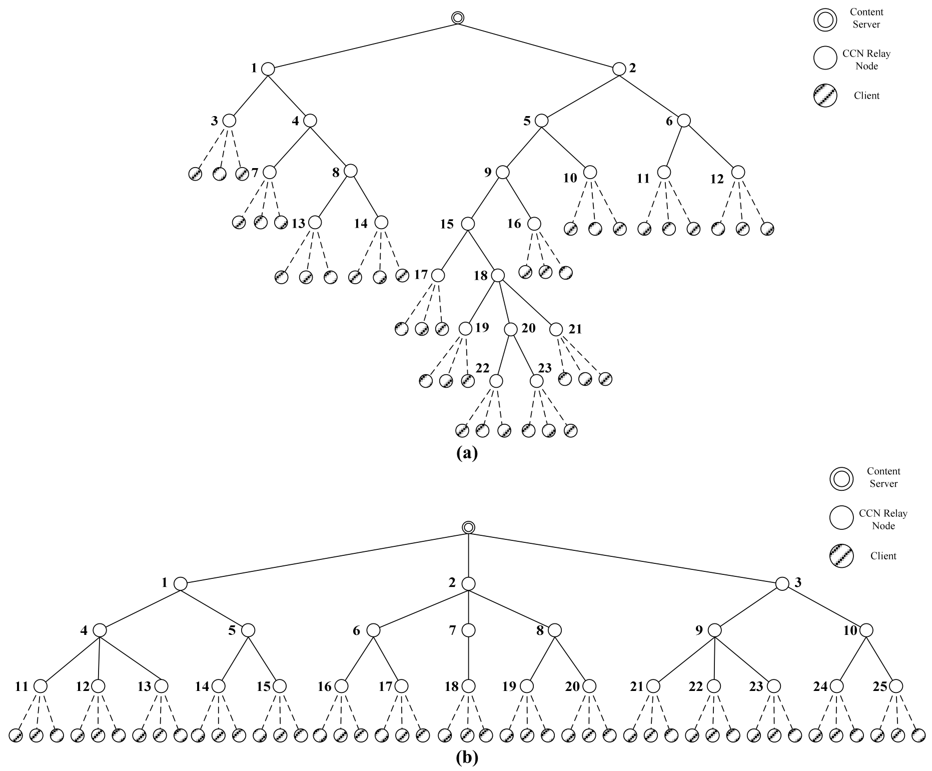


FIGURE 6. Simulation topologies: (a) asymmetric tree topology, (b) symmetric tree topology.

TABLE 2. The total amount of required peak bandwidth for all routes in an asymmetric tree topology.

Time (sec.)	Content Delivery Route													
	Route #1	Route #2	Route #3	Route #4	Route #5	Route #6	Route #7	Route #8	Route #9	Route #10	Route #11	Route #12	Route #13	Total
600	21.69	29.42	37.28	37.28	44.90	46.96	49.03	49.03	46.96	37.16	29.42	29.42	29.42	487.99
1200	17.45	23.07	28.68	28.68	34.30	35.58	16.68	16.68	35.58	28.68	23.07	23.07	23.07	334.61
1800	17.45	23.07	16.58	16.58	34.30	15.40	16.68	16.68	15.40	28.68	23.07	15.49	15.49	254.87
2400	17.45	14.99	16.58	16.58	34.30	15.40	16.68	16.68	15.40	28.68	23.07	15.49	15.49	246.80
3000	13.41	14.99	16.58	16.58	18.16	15.40	16.68	16.68	15.40	28.68	23.07	15.49	15.49	226.62
3600	12.20	13.78	15.36	15.36	18.16	15.40	16.68	16.68	15.40	16.58	23.07	15.49	15.49	209.65
4200	12.20	13.78	12.57	12.57	18.16	15.40	16.68	16.68	15.40	16.58	14.99	14.99	14.99	194.99
4800	12.20	12.55	13.17	13.17	18.16	13.84	15.12	15.12	13.84	15.01	13.43	13.43	13.43	182.47
5400	12.20	12.55	11.35	11.35	15.13	13.85	10.78	10.78	13.85	13.57	12.78	12.78	12.78	163.74
6000	11.07	11.52	12.04	12.04	15.13	9.49	10.78	10.78	9.49	13.57	12.78	13.38	13.38	155.45
6600	11.07	11.42	11.18	11.18	15.13	9.49	10.78	10.78	9.49	13.57	12.78	13.38	13.38	153.64
7200	11.07	11.42	11.18	11.18	15.13	9.49	10.78	10.78	9.49	13.57	12.78	13.38	13.38	153.64
7800	11.07	11.42	11.18	11.18	15.13	9.49	10.78	10.78	9.49	13.57	12.78	13.38	13.38	153.64
8400	11.07	11.42	11.18	11.18	15.13	9.49	10.78	10.78	9.49	13.57	12.78	13.38	13.38	153.64
9000	11.07	11.42	11.18	11.18	15.13	9.49	10.78	10.78	9.49	13.57	12.78	13.38	13.38	153.64

that CPLEX requires approximately an hour to obtain the solution because the number of variables in the problem is approximately 130,000, and thus it is can be used as only a benchmarking for the performance comparison. The comparisons of total amount of required peak bandwidth according to tree network topology and content popularity are presented in Fig. 7. As shown in Fig. 7 (a) and (b), when the video requests follow a Zipf distribution, the total amount of required peak bandwidth of LFU is less than those of LRU and ProbCache because LRU and ProbCache do not consider content popularity. While, as shown in Fig. 7 (c) and (d), when the video requests follow a uniform distribution, ProbCache shows a lower total amount of required peak bandwidth than LFU since LFU frequently replaces cached chunks with a similar video request rate. The proposed caching system provides the lowest total amount of required peak bandwidth compared with existing caching systems, regardless of tree network topology and content popularity. Furthermore, it is

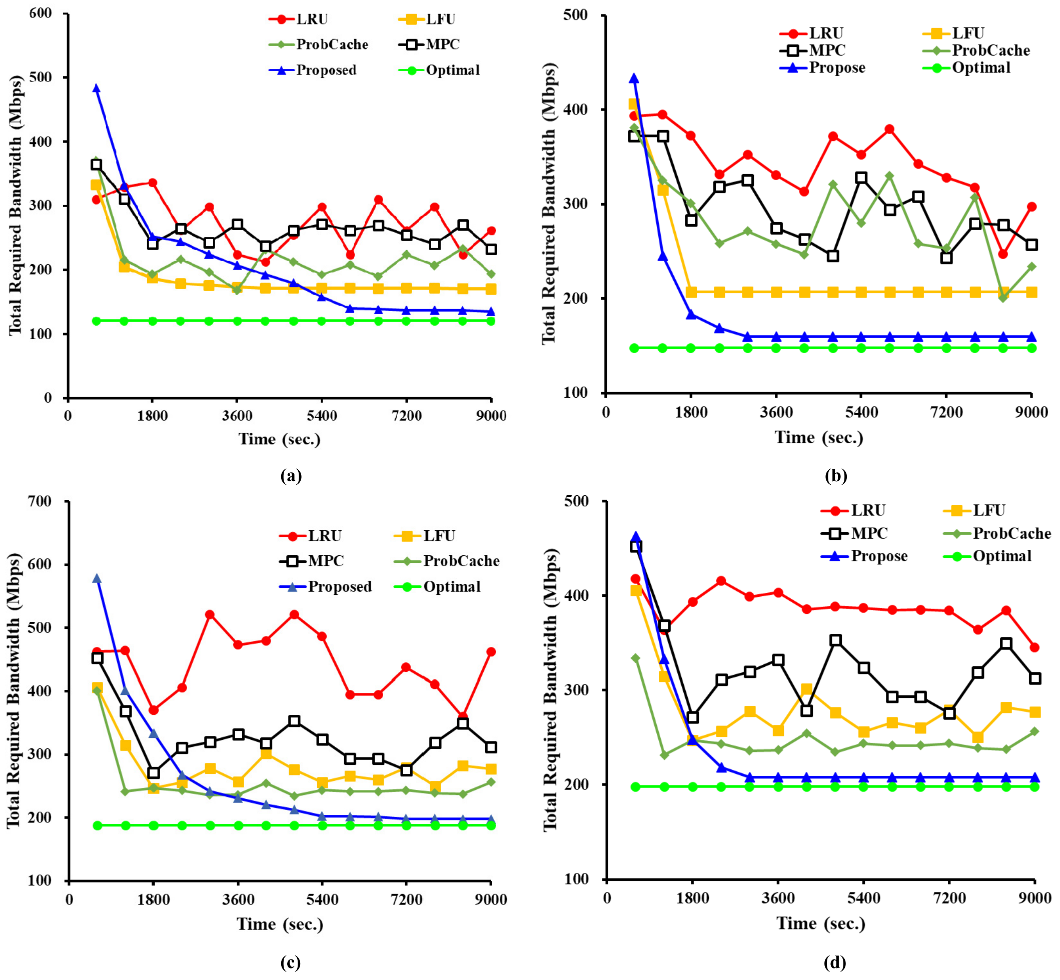


FIGURE 7. **Comparison of the total amount of required peak bandwidth: (a) Asymmetric tree topology & Zipf distribution, (b) Symmetric tree topology & Zipf distribution, (c) Asymmetric tree topology & Uniform distribution, (d) Symmetric tree topology & Uniform distribution.**

apparently observed that the total amount of required peak bandwidth of the proposed caching system monotonically decreases and approaches the optimal solution by CLPEX.

B. PERFORMANCE VERIFICATION OVER REAL CELLULAR WIRELESS NETWORK TESTBED

In this section, we demonstrate the performance of the proposed caching system over a real cellular wireless network testbed. During the experiments, the proposed caching system has been fully implemented by using a Raspberry Pi, C/C++, CCNx, and Ubuntu MATE 16.04 [31]. Fig.8 (a) presents the network topology of the real cellular wireless network testbed that is constructed to emulate a cellular network such as a Long-Term Evolution (LTE) [29] and an LTE-Advanced (LTE-A) [33]. Fig. 8 (b) shows the web interface of the CCNx that represents cache statistics and the forwarding table at each node. The blue box indicates the activated interface. Therefore, the interest and data packets are delivered via UE1, eNB1, S-GW1, and the Content Server. Fig. 9 shows the location of the cellular network components in our laboratory.

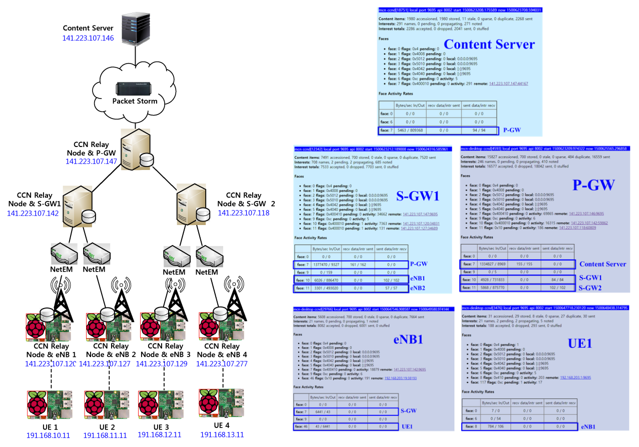


FIGURE 8. **Network topology for the real cellular wireless network testbed.**

In addition, the network emulator (NetEM) [28] is employed to emulate a cellular network environment and generate background traffic to limit the available bandwidth to 2.5Mbps. The video requests follow a uniform distribution.

TABLE 3. Changes in the highest and the lowest priority of the cached chunks.

Time (sec.)	Video	eNB1		S-GW1		P-GW	
		Highest Priority	Lowest Priority	Highest Priority	Lowest Priority	Highest Priority	Lowest Priority
0	Elephant-Dream	-	-	-	-	-	-
	Of-Forests-And-Man	-	-	-	-	-	-
	RedBull-Playstreet	-	-	-	-	-	-
600	Elephant-Dream	1	662	-	-	-	-
	Of-Forests-And-Man	-	-	-	-	-	-
	RedBull-Playstreet	1	118	-	-	-	-
1200	Elephant-Dream	1	662	-	-	-	-
	Of-Forests-And-Man	-	-	1	237	-	-
	RedBull-Playstreet	1	118	119	582	-	-
1800	Elephant-Dream	1	662	-	-	-	-
	Of-Forests-And-Man	-	-	1	237	238	635
	RedBull-Playstreet	1	118	119	582	-	-
2400	Elephant-Dream	1	662	-	-	-	-
	Of-Forests-And-Man	-	-	1	237	238	635
	RedBull-Playstreet	1	118	119	582	-	-

TABLE 4. Summary of performance comparison with existing caching system.

System	Avg. Number of buffer underflows	Avg. Buffering duration (sec.)
LRU	1.69	4.72
LFU	2.31	5.94
ProbCache	2.42	4.25
Proposed	1.39	4.06

The changes in the highest and lowest caching priorities of the proposed caching system are represented in Table 3. In the proposed caching system, eNB1 firstly caches the chunks of "Elephants-Dream" because the video bitrates of "Elephants-Dream" are somewhat higher than those of "Of-Forests-And-Man" and "RedBull-Playstreet". The chunks of the "Of-Forests-And-Man" and "RedBull-Playstreet" are partially cached in eNB1, S-GW1, and P-GW. Cached chunks are replaced every 600 s until the required peak bandwidth converges. The proposed caching system gradually reduces the required peak bandwidth as shown in Fig. 10. Furthermore, it is shown that the proposed caching system has a lower required peak bandwidth than the existing caching systems and presents almost the same performance as CPLEX after converging.

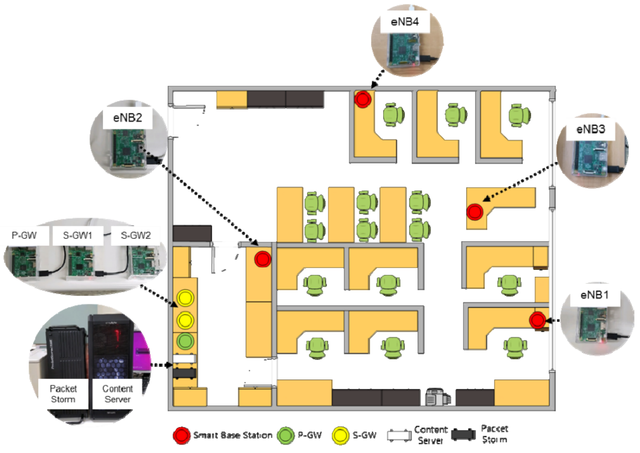


FIGURE 9. Locations of cellular network components in the laboratory.

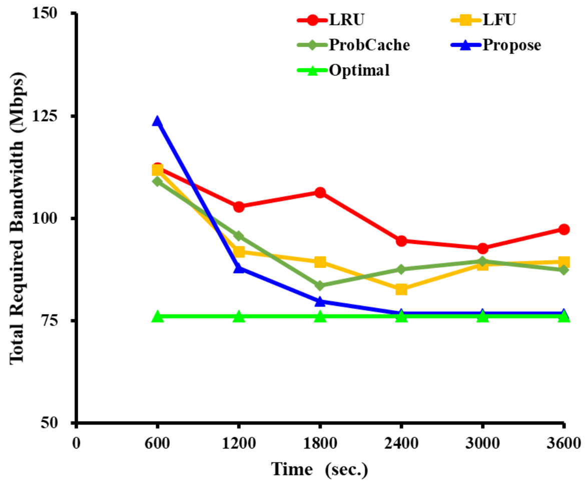


FIGURE 10. Comparison of the total amount of required peak bandwidth in the real cellular wireless network testbed.

Finally, we compare the temporal video quality among the proposed caching system and existing caching systems. The number of buffer underflows and the buffering duration incurring unsmooth video playout are adopted as performance measures for temporal video quality. Figs. 11 and 12 provide the cumulative curves of received and consumed video data at UE #1. It is observed that the proposed caching system provides seamless video streaming for "Elephants-Dream" as all its chunks are cached at eNB1 due to the higher bitrate. When the test video sequence is "Of-Forests-And-Man.", the proposed caching system still achieves the lowest number of buffer underflows compared with other existing caching systems as shown in Fig. 12. However, frozen video is sometimes detected since its chunks are partially cached at P-GW1 and S-GW1 in the proposed caching system. The average number of underflows and average buffering duration are summarized in Table 4. It is observed in the table that the proposed caching system provides the lowest average number of underflows and buffering duration compared with other existing

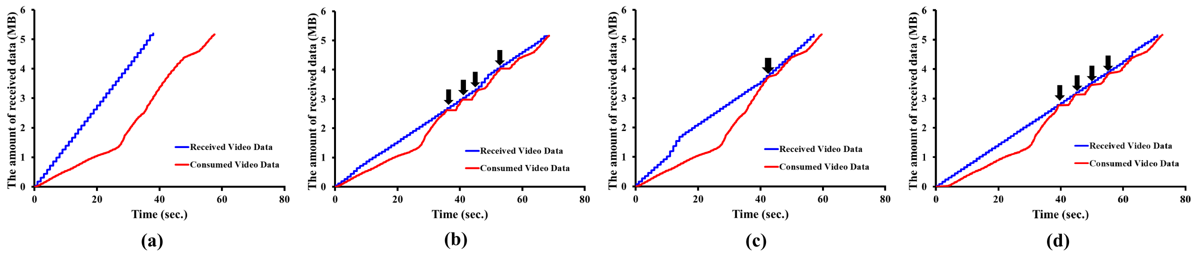


FIGURE 11. Cumulative curves of the received video data and consumed video data in the case of "Of-Forests-And-Man": (a) Proposed caching system, (b) LRU, (c) LFU and (d) ProbCache (The black arrows indicate a buffer underflow).

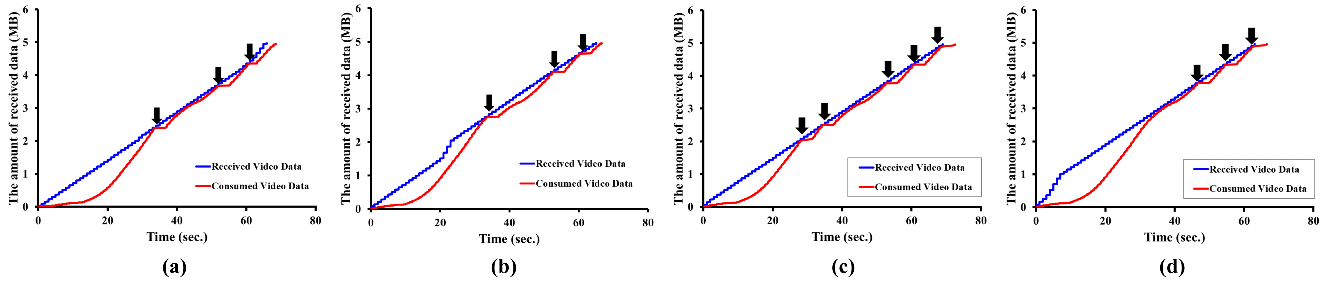


FIGURE 12. Cumulative curves of the received video data and consumed video data in the case of "Elephants-Dream": (a) Proposed caching system, (b) LRU, (c) LFU and (d) ProbCache (The black arrows indicate a buffer underflow).

caching systems. Consequently, the proposed caching system provides streaming services while significantly reducing buffer underflow and buffering duration.

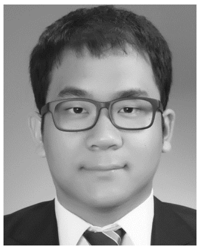
IV. CONCLUSION

In this paper, we have proposed a metafile based progressive caching system for seamless streaming services with high network utilization over the CCN tree. The proposed caching system considers caching range and content popularity to improve network utilization. First of all, we construct a metafile for the CCN to provide the cached chunk priority. Next, a caching information exchanging protocol is designed to efficiently deliver the caching information among CCN nodes. Based on this information, the proposed caching system determines the caching range of video data to minimize the required peak bandwidth for all links. The proposed caching system is examined by using not only simulations for various large-scale network environments, but also a real cellular wireless network testbed. Based on the simulations and experimental results, we have shown that the proposed caching system can support better video streaming services with reduced network stress.

REFERENCES

[1] Cisco Visual Networking Index, *Global Mobile Data Traffic Forecast Update, 2016–2021*, document C11-481360-01, Cisco Systems, San Jose, CA, USA, 2017, pp. 1–17.
[2] H. Che, Z. Wang, and Y. Tung, "Analysis and design of hierarchical Web caching systems," in *Proc. IEEE INFOCOM*, Apr. 2001, pp. 1416–1424.
[3] Y. Chen, L. Qiu, W. Chen, L. Nguyen, and R. H. Katz, "Efficient and adaptive Web replication using content clustering," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 6, pp. 979–994, Aug. 2003.
[4] A. Rowstron and P. Druschel, "Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility," *SIGOPS Oper. Syst. Rev.*, vol. 35, no. 5, pp. 188–201, Oct. 2001.
[5] V. Gopalakrishnan, B. Silaghi, B. Bhattacharjee, and P. Keleher, "Adaptive replication in peer-to-peer systems," in *Proc. 24th Int. Conf. Distrib. Comput. Syst.*, 2004, pp. 360–369.
[6] H. Shen, "An efficient and adaptive decentralized file replication algorithm in P2P file sharing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 6, pp. 827–840, Jun. 2010.
[7] J. Wang, "A survey of Web caching schemes for the Internet," *SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 5, pp. 36–46, Oct. 1999.
[8] A. Vakali and G. Pallis, "Content delivery networks: Status and trends," *IEEE Internet Comput.*, vol. 7, no. 6, pp. 68–74, Nov. 2003.
[9] S. Androutsellis-Theotokis and D. Spinellis, "A survey of peer-to-peer content distribution technologies," *ACM Comput. Surv.*, vol. 36, no. 4, pp. 335–371, Dec. 2004.
[10] A. Ioannou and S. Weber, "A survey of caching policies and forwarding mechanisms in information-centric networking," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 4, pp. 2847–2886, 4th Quart., 2016.
[11] A. Passarella, "A survey on content-centric technologies for the current Internet: CDN and P2P solutions," *Comput. Commun.*, vol. 35, no. 1, pp. 1–32, 2012.
[12] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. Briggs, and R. L. Braynard, "Networking named content," in *Proc. ACM CoNEXT*, 2009, pp. 1–12.
[13] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proc. 2nd Ed. ICN Workshop Inf.-Centric Netw.*, 2012, pp. 55–60.
[14] C. Bernardini, T. Silverston, and O. Festor, "MPC: Popularity-based caching strategy for content centric networks," in *Proc. IEEE ICC*, Jun. 2013, pp. 3619–3623.
[15] K. Cho, M. Lee, K. Park, T. T. Kwon, Y. Choi, and S. Pack, "Wave: Popularity-based and collaborative in-network caching for content-oriented networks," in *Proc. IEEE INFOCOM WKSHPS*, Mar. 2012, pp. 316–321.
[16] K. Suksomboon *et al.*, "PopCache: Cache more or less based on content popularity for information-centric networking," in *Proc. IEEE Conf. Local Comput. Netw. (LCN)*, Oct. 2013, pp. 236–243.
[17] J. Ren *et al.*, "MAGIC: A distributed MAx-gain in-network caching strategy in information-centric networks," in *Proc. IEEE Conf. INFOCOM Workshop*, Apr./May 2014, pp. 470–475.

[18] Y. Wang, Z. Li, G. Tyson, S. Uhlig, and G. Xie, "Design and evaluation of the optimal cache allocation for content-centric networking," *IEEE Trans. Comput.*, vol. 65, no. 1, pp. 95–107, Jan. 2016.
[19] J. Sung, M. Kim, K. Lim, and J.-K. K. Rhee, "Efficient cache placement strategy in two-tier wireless content delivery network," *IEEE Trans. Multimedia*, vol. 18, no. 6, pp. 1163–1174, Jun. 2016.
[20] C. Li, L. Toni, J. Zou, H. Xiong, and P. Frossard, "QoE-driven mobile edge caching placement for adaptive video streaming," *IEEE Trans. Multimedia*, vol. 30, no. 4, pp. 965–984, Apr. 2018.
[21] Z. Su, Q. Xu, F. Hou, Q. Yang, and Q. Qi, "Edge caching for layered video contents in mobile social networks," *IEEE Trans. Multimedia*, vol. 19, no. 10, pp. 2210–2221, Oct. 2017.
[22] Z. Miao and A. Ortega, "Scalable proxy caching of video under storage constraints," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 7, pp. 1315–1327, Sep. 2002.
[23] H. R. Oh and H. Song, "Metafile-based scalable caching and dynamic replacing algorithms for multiple videos over quality-of-service networks," *IEEE Trans. Multimedia*, vol. 9, no. 7, pp. 1535–1542, Nov. 2007.
[24] S. Williams, M. Abrams, C. R. Standridge, G. Abdulla, and E. A. Fox, "Removal policies in network caches for World-Wide Web documents," in *Proc. ACM SIGCOMM*, Aug. 1996, pp. 293–305.
[25] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache 'less for more' in information-centric networks (extended version)," *Computer Commun.*, vol. 36, no. 7, pp. 758–770, Apr. 2013.
[26] J. T. Robinson and M. V. Devarakonda, "Data cache management using frequency-based replacement," in *Proc. ACM SIGMETRICS Conf. Meas. Modeling Comput. Syst.*, May 1990, pp. 134–142.
[27] A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM:NDN simulator for NS-3," Univ. California, Los Angeles, CA, USA, NDN Project, Tech. Rep. NDN-0005, Jul. 2012.
[28] Stephen Hemminger. (2005). *Network Emulation With NetEm*. [Online]. Available: https://linux.conf.au/
[29] *Raspberry Pi 3*. Accessed: Apr. 9, 2019. [Online]. Available: https://www.raspberrypi.org
[30] *CCNx*. Accessed: Apr. 9, 2019. [Online]. Available: https://github.com/ProjectCCNx/ccnx
[31] *Ubuntu Mate*. Accessed: Apr. 9, 2019. [Online]. Available: https://ubuntu-mate.org/
[32] D. Astely, E. Dahlman, A. Furuskär, Y. Jading, M. Lindström, and S. Parkvall, "LTE: The evolution of mobile broadband," *IEEE Commun. Mag.*, vol. 47, no. 4, pp. 44–51, Apr. 2009.
[33] *Technical Specification Group Radio Access Network; Requirements for Further Advancements for Evolved Universal Terrestrial Radio Access (E-UTRA) (LTE-Advanced) (Release 12)*, document TR 36.913 v12.0.0, 3GPP, Sophia Antipolis, France, Sep. 2014.
[34] *CPLEX*. Accessed: Apr. 9, 2019. [Online]. Available: https://www.ibm.com/analytics/cplex-optimizer



HYUNMIN NOH received the B.S. degree from the School of Computer Science and Engineering, Hanyang University, South Korea, in 2015. He is currently pursuing the Ph.D. degree in computer science and engineering with the Pohang University of Science and Technology (POSTECH), South Korea. His research interests are in the areas of content centric networking, software-defined networking, and multimedia networking.

HWANGJUN SONG received the B.S. and M.S. degrees from the Department of Control and Instrumentation (EE), Seoul National University, South Korea, in 1990 and 1992, respectively, and the Ph.D. degree in electrical engineering-systems from the University of Southern California, Los Angeles, CA, USA, in 1999. From 2000 to 2005, he was an Assistant Professor/Vice Dean of admission affairs with Hongik University, Seoul, South Korea. Since 2005, he has been with the Department of Computer Science and Engineering, Pohang University of Science and Technology (POSTECH), South Korea. He was a courtesy Associate Professor with the University of Florida, on sabbatical leave (2011–2012), and has served as the Vice President of the Korean Institute of Information Scientists and Engineers (2016–2017). He was an APSIPA Distinguished Lecturer (2017–2018). His research interests include multimedia signal processing and communication, image/video compression, digital signal processing, network protocols necessary to implement functional image/video applications, and network control systems. He is an Editorial Board Member of the *Journal of Visual Communication and Image Representation* and an Associate Editor of the *Journal of Communications and Networks*. He has served as an Editorial Board Member of the *International Journal of Vehicular Technology* and a Guest Editor of the Special Issue on Network Technologies for Emerging Broadband Multimedia Services in the *Journal of Visual Communication and Image Representation* and the Special Issue on Wireless and Mobile Networks in the *International Journal of Ad Hoc and Ubiquitous Computing*.