sensors

MDPI

Article

An Efficient Coded Streaming Using Clients' Cache

Tae-Won Ban †, Woongsup Lee and Jongyeol Ryu *

Department of Information and Communication Engineering, Gyeongsang National University, Tongyeong 53064, Korea; twban35@gnu.ac.kr (T.-W.B.); wslee@gnu.ac.kr (W.L.)

* Correspondence: jongyeol_ryu@gnu.ac.kr

† Current address: Marine Science Bldg 807, Tongyeong, Gyeongnam, Korea.

Received: 25 September 2020; Accepted: 30 October 2020; Published: 31 October 2020



Abstract: As multimedia traffic has been increasing and is expected to grow more sharply, various technologies using caches have been attracting lots of attention. As one breakthrough technology to deal with the explosively growing traffic, exclusive OR (XOR)-based index coding has been widely investigated because it can greatly enhance the efficiency of network resource by reducing the number of transmissions. In this paper, we investigate how to apply XOR-based index coding to large-scaled practical streaming systems for video traffic that accounts for more than 70% of total Internet traffic. Contrary to most previous studies that have focused on theoretical analysis of optimal performance or development of optimal index coding schemes, our study proposes a new XOR coding-based video streaming (XC). We also propose a new grouping algorithm for creating XC groups while guaranteeing the complete backward compatibility of XC with existing streaming schemes such as unicast (UC), multicast (MC), and broadcast (BC). The performance of the proposed scheme is analyzed in two steps. First, the behavior of video contents in caches at clients is modeled as a Markov chain, and the steady-state probabilities and caching probabilities for each piece of video content are derived. Based on the probabilities, the performance of the proposed system is then analyzed in terms of the average number of connections that each client requires in order to receive one video content. Our numerical results show that the proposed video streaming scheme using XC can reduce the average number of transmissions by up to 18%, compared to the conventional scheme.

Keywords: edge caching; streaming; multimedia; coded streaming; steady-state probability

1. Introduction

A recent study showed that the total amount of Internet traffic will increase three-fold over five years from 2017 to 2022, and video traffic would be growing more sharply than other types of traffic [1]. More specifically, it was predicted that the video traffic that accounted for about 75% of total Internet traffic in 2017 would increase to about 82% in 2022. It was also revealed by another interesting study that the most popular 50 videos accounted for almost 80% of the total number of views in a major over-the-top (OTT) service provider, YouTube [2]. Video service providers deploy huge content delivery systems and manage their own systems to meet the increasing demand, but the approach causes a tremendous cost to the companies [3]. An alternative is to use caching [4,5]. If a requested content exists in the cache of a nearby proxy server, clients can receive the cached copy, which typically reduces delivery time. Recently, the concept of caching has been extended to wireless multimedia streaming systems to cope with the sharply increasing data traffic by offloading increasing video traffic or improving the spectral efficiency of wireless networks [6–15]. The problem of caching placement was investigated, and some state-of-the-art solutions were introduced in [6–9]. A dynamic deep reinforcement learning-based management framework for virtual cache slicing was proposed to manage the limited cache resources when service providers share a common physical infrastructure [6].

A probabilistic caching placement was investigated to control cache-based channel selection diversity and network interference in a stochastic wireless caching helper network [7]. A tradeoff between the content diversity gain and the cooperative gain according to content placements was investigated and a probabilistic content placement to optimally balance the tradeoff was proposed [8]. Device-to-device (D2D) caching network, where each node caches some contents and another node can receive these contents directly from its nearby nodes instead of a remote service provider, was investigated in [10,11].

Contrary to previous studies that considered transmitters with cache and studied how to use the transmitters' cache efficiently, some studies focused on clients with cache [12–19]. If a client wants to play a content which has been stored in its cache, the content can be played directly from the cache with no outbound connection. A joint problem of content pushing and recommendation for cache-enabled mobile users was formulated and a reinforcement learning-based framework for maximizing the profit of mobile network operators was proposed in [12]. Novel transmission techniques based on state-of-the-art interference management schemes such as interference cancellation, zero-forcing, and interference alignment were proposed for cache-aided radio access networks where edge nodes and user equipment are both equipped with cache [13]. An erasure broadcast network was investigated considering two disjoint sets of receivers: a set of weak receivers with an equal size of cache and a set of strong receivers with no cache [14]. It was proposed to exploit limited cached packets as side information to cancel incoming interference at the receiver side in a stochastic network where the random locations of base stations and users are modeled by Poisson point process [15].

Moreover, more advanced transmission schemes that use the contents stored in clients' cache more efficiently were proposed to improve the efficiency of limited network resource [16–19]. If a transmitter has the perfect information of contents stored in clients' caches, the transmitter can transmit multiple contents requested by different clients at once by using XOR index coding and each client can recover its content by performing XOR operations of the XOR coded data and contents stored in its cache, which can enhance the efficiency of network resource by reducing the number of transmissions[16]. The exact broadcast rate which had remained unknown was analyzed using linear programs [17]. A simple inner bound was established on the general index coding problem and it was demonstrated that the inner bound is tight for all index coding problems of up to five messages [18]. A pruning algorithm that can find the optimal clique cover index code and transmission time allocation for minimizing wireless outage probability was proposed to reduce a computational complexity of a brute-force searching algorithm [19]. Most of aforementioned studies were devoted to theoretical research to analyze optimal performance or to develop optimal coding schemes, and they are not thus sufficient to be applied to real large-scaled video streaming systems. In addition, the real advantages of the index coding-based video streaming and its compatibility with existing streaming schemes have never been verified.

In this paper, we thus propose a new video streaming system that supports XC. We also propose a new grouping algorithm for creating XC groups while guaranteeing the complete backward compatibility of XC with existing streaming schemes such as UC, MC, and BC. The performance of the proposed system is analyzed and compared with that of a conventional streaming system to verify its gain. We first model the behavior of contents in caches at clients as a Markov chain and derive the steady-state probabilities and caching probabilities for each video to mathematically analyze the statistical characteristics of caches at clients in video streaming systems. Based on the statistical characteristics of caches, the performance of the proposed system is analyzed in terms of the average number of connections that each client requires in order to receive one video content. The proposed scheme can be widely applied for various types of networks including mobile networks and sensor networks with video streaming.

The rest of this paper is organized as follows. In Section 2, a video streaming system where clients are equipped with cache, and we derive the steady-state probabilities and caching probabilities for each video based on a Markov chain. A new video streaming system for supporting XC is proposed and a grouping algorithm for the proposed streaming system is also described in Section 3. Numerical results are shown in Section 4, and the conclusions of this paper are drawn in Section 5.

2. A Streaming System Using Clients' Cache

We investigate a video streaming system with a single streaming server and N clients, which is illustrated in Figure 1. The streaming server and each client are all equipped with cache that can store up to V and C videos ($V > C$), respectively. It is assumed that all videos have the same size. $\mathbb{C}_n$ denotes the cache of client n and $|\mathbb{C}_n| = C \; \forall \; n$. If each client plays video contents independently of others, the statistical characteristics of cache at each client will be identical and hence, we omit the index n for a simplicity of the notation, now and hereafter. $\mathbb{C}(s)$ denotes the video that is stored in the s-th memory of a cache $\mathbb{C}$, where $1 \leq s \leq C$. Thus, $\mathbb{C}(1)$ and $\mathbb{C}(C)$ indicate the most recently used video and the least recently used video, respectively. $\mathbb{L}_s$ denotes a set of videos used later than $\mathbb{C}(s)$ and is given by

$$\mathbb{L}_s = \begin{cases} \{\mathbb{C}(1), \cdots, \mathbb{C}(s-1)\}, & \text{if } s \neq 1 \\ \emptyset, & \text{if } s = 1. \end{cases} \tag{1}$$

$\mathbb{R}_s$ denotes a set of videos used earlier than $\mathbb{C}(s)$ and is given by

$$\mathbb{R}_s = \begin{cases} \{\mathbb{C}(s+1), \cdots, \mathbb{C}(C)\}, & \text{if } s \neq C \\ \emptyset, & \text{if } s = C, \end{cases} \tag{2}$$

and can be also obtained by $\mathbb{R}_s = \mathbb{C} \setminus \mathbb{L}_s \setminus \{\mathbb{C}(s)\} = \mathbb{C} \setminus \{\mathbb{C}(1), \cdots, \mathbb{C}(s)\}$.

The normalized popularity of a video that ranks v-th can be predicted by Zipf distribution [20,21], which is given by

$$f_v = \frac{1/v^{\alpha}}{\sum_{n=1}^{V} (1/n^{\alpha})}, \; 1 \leq v \leq V, \tag{3}$$

where f_v satisfies $\sum_{v=1}^{V} f_v = 1$ and α is the exponent characterizing Zipf distribution [22]. Zipf distribution with $\alpha = 0$ turns into a uniform distribution, by which all V videos have the same normalized popularity, $\frac{1}{V}$. We use the least recently used (LRU) algorithm as a cache replacement strategy [23], by which clients' caches are autonomously updated without consuming extra network resource after streaming a video content. If a client wants to play a video content v that is not stored in the cache, which denotes $v \notin \mathbb{C}$, the content v will be streamed by one of MC, XC, or UC. After playing the content v, LRU discards $\mathbb{C}(C)$, which is the least recently used one, and the content v will be stored in the first place of the cache after shifting all contents in the cache to the right. Thus, $\mathbb{C}$ will be updated as $\mathbb{C} = \{v, \mathbb{C}(1), \cdots, \mathbb{C}(C-1)\}$. On the other hand, if a client wants to play a content v stored in the s-th place of the cache, which denotes $v = \mathbb{C}(s)$, $1 \leq s \leq C$, the content v will be directly played by local playing (LC) from the cache without using any network resource for streaming. Then, the content v will be re-located to the first place of the cache to ensure that the most recently used content will stay in the cache the longest. The contents in $\mathbb{L}_s$ will be only shifted to the right, while there will be no change on contents in $\mathbb{R}_s$. Thus, $\mathbb{C}$ will be updated as $\mathbb{C} = \{v, \mathbb{C}(1), \cdots, \mathbb{C}(s-1), \mathbb{C}(s+1), \cdots, \mathbb{C}(C)\}$. The case of $s = 1$ denotes that the same content has been being played twice in a row and there is then no change on the cache.

A certain content k $(1 \leq k \leq V)$ might be stored in the client's cache or might not be stored. In the case that content k is stored in the cache, the content k can have C states based on its stored location. Thus, each content k can have total $(C+1)$ states in $\mathbb{C}$. The $(C+1)$-th state denotes that the content is not stored in the cache. The behavior of content k in the cache can be modeled by a Markov chain, as depicted in Figure 2. Transition probability p_k^{sj} denotes a probability that a content k staying at state s moves to state j after a client plays the content v.

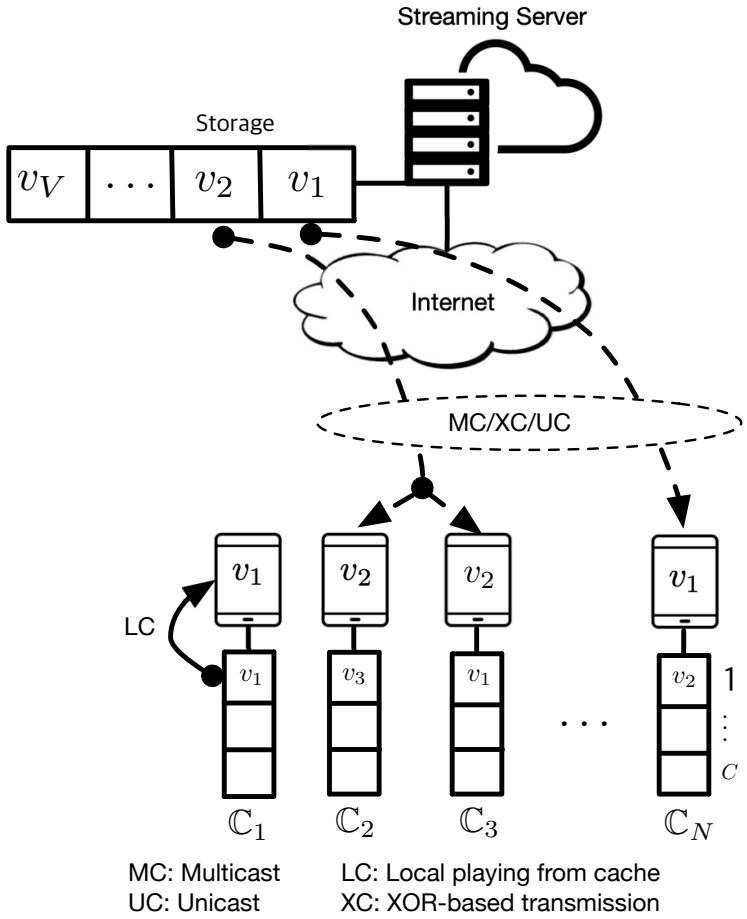


Figure 1. A video streaming system where each client is equipped with a cache size C.

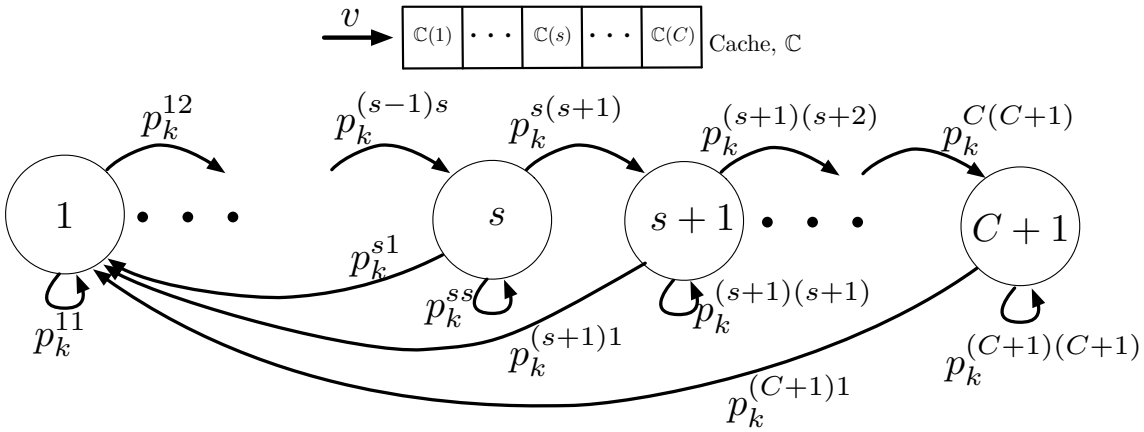


Figure 2. A state transition diagram of a video k in a cache with a length of C.

For the case of $s = 1$, a content k staying in the first place has two possibilities of transition. If $v = k$, which means that a client plays the content k by using LC, the video k will be still staying in the first place with no transition. Otherwise, it will move to the second place. Two transition probabilities are thus given as

$$p_k^{11} = \Pr[v = k] = f_k \tag{4}$$
$$p_k^{12} = \Pr[v \neq k] = 1 - p_k^{11} = 1 - f_k. \tag{5}$$

For the case of $2 \leq s \leq C$, a content k has three possibilities of transition. If $v = k$, then the content k will be played by LC and move to the first place after being played. If $v \in \mathbb{L}_s$, the content k will have no transition. Finally, if $v \in \mathbb{R}_s$ or $v \notin \mathbb{C}$, the content k will be shifted to the right and thus move to the $(s+1)$-th place. The corresponding transition probabilities are given as

$$p_k^{s1} = \Pr[v = k] = f_k \tag{6}$$
$$p_k^{ss} = \Pr[v \in \mathbb{L}_s] \tag{7}$$
$$= \sum_{X \in {}_{\tilde{\mathbb{V}}_k}\mathbb{P}_{s-1}} \frac{\prod\limits_{x \in X} f_x}{\sum_{X \in {}_{\tilde{\mathbb{V}}_k}\mathbb{P}_{s-1}} \prod\limits_{x \in X} f_x} \sum_{x \in X} f_x \tag{8}$$
$$p_k^{s(s+1)} = \Pr\left[v \in \mathbb{R}_s \text{ or } v \notin \mathbb{C}\right] = 1 - p_k^{s1} - p_k^{ss}, \tag{9}$$

where $\tilde{\mathbb{V}}_k$ denoting the index set of contents except for k can be obtained by

$$\tilde{\mathbb{V}}_k = \mathbb{V} \setminus \{k\} = \{1, 2, \cdots, k-1, k+1, \cdots, V\}, \tag{10}$$

and ${}_{\mathbb{X}}\mathbb{P}_n$ denotes a set that consists of all possible permutations of choosing n objects from a set $\mathbb{X}$. Thus, ${}_{\tilde{\mathbb{V}}_k}\mathbb{P}_{s-1}$ denoting a set of all possible $\frac{(V-1)!}{(V-s)!}$ permutations of $(s-1)$ contents that can be chosen from the set $\tilde{\mathbb{V}}_k$ represents all possible combinations of $\mathbb{L}_s$. For the case of $s = C+1$, the content k has two cases of transition. If $v = k$, the content k move to the first place after being played. Otherwise, the content k will be discarded from the cache. The corresponding transition probabilities are given as

$$p_k^{(C+1)1} = \Pr[v = k] = f_k \tag{11}$$
$$p_k^{(C+1)(C+1)} = \Pr[v \neq k] = 1 - p_k^{(C+1)1} = 1 - f_k. \tag{12}$$

A square matrix consisting of state transition probabilities for the content k is defined as

$$\mathbf{P}_k \triangleq [p_k^{ij}],\ 1 \leq i \leq C+1,\ 1 \leq j \leq C+1, \tag{13}$$

the size of which is $(C+1) \times (C+1)$. Let s_i be a steady-state probability that the content k is staying in state i at an arbitrary moment. Then, $\mathbf{s}_k = [s_1, s_2, \cdots, s_{C+1}]$ denoting the vector consisting of steady-state probabilities for all states for the content k can be obtained by solving the following equation [24]:

$$\begin{aligned} &\mathbf{s}_k \times \mathbf{P}_k = \mathbf{s}_k \\ &\text{subject to} \sum_{i=1}^{C+1} s_i = 1. \end{aligned} \tag{14}$$

The solution of (14) can be obtained by several approaches and we, in this paper, use a pseudo inverse-based approach [24]. The equation in (14) can be rewritten as

$$\begin{aligned} \mathbf{s}_k \times \left[\mathbf{P}_k - \mathbf{I}\right] &= \mathbf{0} \\ \text{subject to } \textstyle\sum_{i=1}^{C+1} s_i &= 1, \end{aligned} \tag{15}$$

where $\mathbf{I}$ and $\mathbf{0}$ denote an identity matrix with the same size as $\mathbf{P}_k$ and a row vector with $(C+1)$ zeros, respectively. The constraint in (15) can be incorporated into the equation as

$$\mathbf{s}_k \times \left[(\mathbf{P}_k - \mathbf{I})|\mathbf{1}\right] = \left[\mathbf{0}|1\right], \tag{16}$$

by appending $\mathbf{1}$, which is a column vector with $(C+1)$ ones, to the right of the matrix $(\mathbf{P}_i - \mathbf{I})$ and 1 to the right of the vector $\mathbf{0}$, respectively. Let $\left[(\mathbf{P}_k - \mathbf{I})|\mathbf{1}\right] = \mathbf{Q}$ and $\left[\mathbf{0}|1\right] = \mathbf{b}$, then (16) can be rewritten as $\mathbf{s}_k\mathbf{Q} = \mathbf{b}$ and $\mathbf{s}_k$ can be obtained as

$$\mathbf{s}_k = \mathbf{b}\mathbf{Q}^T \left(\mathbf{Q}\mathbf{Q}^T\right)^{-1}. \tag{17}$$

When a client wants to play the video content k at an arbitrary moment, the probability that the content is stored in the cache is calculated as

$$c_k = \sum_{i=1}^{C} s_i, \tag{18}$$

while the probability that the content k is not stored in the cache at the arbitrary moment is given as s_{C+1} that can be calculated by $s_{C+1} = 1 - c_k$.

Figure 3 shows probabilities c_k's for various k values, $C \in \{1,2,3\}$, $V = 10$, and $\alpha \in \{0,1,1.5\}$. For $C = 1$, c_k's are the same as normalized popularity values given by Zipf distribution because the current content stored in a cache is the content that a client requested according to Zipf distribution. As C increases, all c_k's increase regardless of the ranks of contents, k. As explained in (3), $\alpha = 0$ makes all contents have the same popularity and c_k's are thus all the same for all k's. As α increases, c_k's of upper rank contents increases while c_k's of lower rank contents decrease.

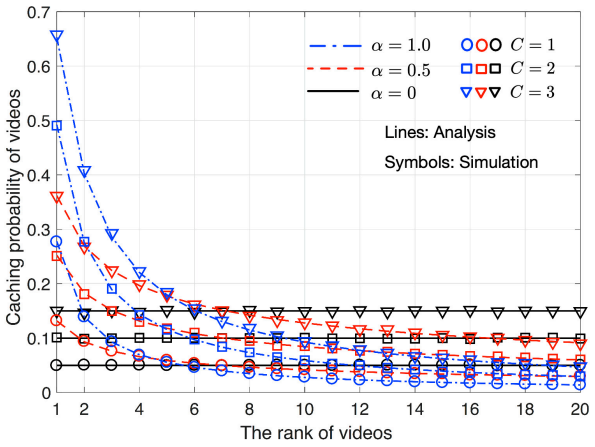


Figure 3. Probability that each content is stored in the cache according to its rank. $C \in \{1,2,3\}$, $V = 10$, and $\alpha \in \{0,1,1.5\}$.

3. Proposed Streaming Scheme Using XOR-Based Coding

The XOR operation has the zero-identity, by which $\mathbf{A} \oplus \mathbf{0} = \mathbf{0} \oplus \mathbf{A} = \mathbf{A}$ is satisfied for a given bit stream $\mathbf{A}$, where $\mathbf{0}$ is the bit stream consisting of 0 and it has the same length as $\mathbf{A}$. It also has the property of self-inverse satisfying $\mathbf{A} \oplus \mathbf{A} = \mathbf{0}$. In addition, XOR is not only commutative, but also associative. Figure 4 illustrates a simplified concept of XC which is a video streaming scheme using XOR coding based on the four aforementioned properties. Client 1 wants to stream v_1 with v_2 cached in $\mathbb{C}_1$, and client 2 wants to stream v_2 with v_1 cached in $\mathbb{C}_2$. Contrary to a conventional scheme where a streaming server transmits v_1 and v_2 by UC using two separate connections, XC generates the coded bit stream $v_1 \oplus v_2$ and transmits it by using a single connection. Then, client 1 and 2 reconstruct v_1 and v_2 from $v_1 \oplus v_2$ by using their cached data as

$$\begin{aligned}(v_1 \oplus v_2) \oplus v_2 &= v_1 \oplus (v_2 \oplus v_2) = v_1 \oplus \mathbf{0} = v_1 \\ (v_1 \oplus v_2) \oplus v_1 &= (v_2 \oplus v_1) \oplus v_1 = v_2 \oplus (v_1 \oplus v_1) = v_2 \oplus \mathbf{0} = v_2,\end{aligned} \tag{19}$$

respectively. In this section, we propose a new transmission algorithm for supporting XC while guaranteeing a backward compatibility with conventional streaming schemes such as LC, UC, MC, and BC, that have been widely used to transmit video contents.

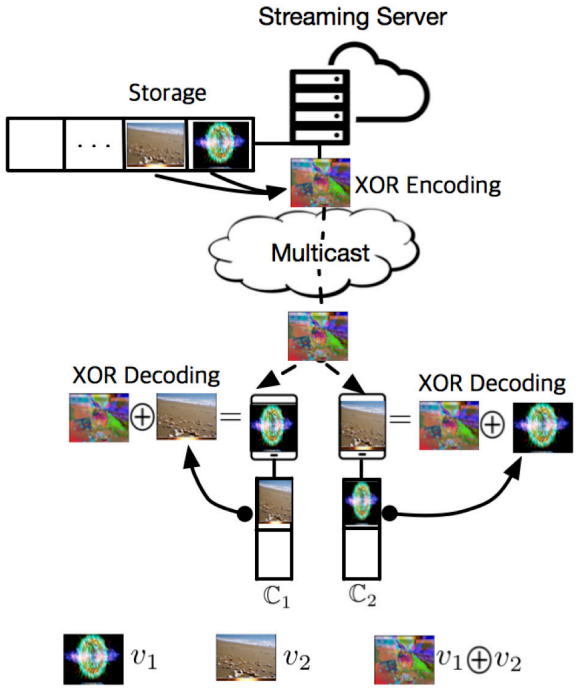


Figure 4. A conceptual illustration of video streaming using XOR-based coding (XC).

3.1. A Case Study with Two Clients ($N = 2$)

To clearly explain the proposed transmission scheme, we first consider a special case with two clients, i.e., $N = 2$, and the proposed transmission scheme will be then extended to the general case to support multiple clients, i.e., $N \geq 2$, in the following subsection. Suppose that clients 1 and 2 want to play $v_1 = i$ and $v_2 = j$, respectively. Each client first checks if the content to play is stored in its cache before requesting the content to a streaming server. If the content is stored in its cache, the client plays the content directly from the cache by LC without an outbound connection. Otherwise, the client transmits a request message including the information of the content to request and the contents of its own cache to a streaming server. Let $\rho_{i,j}^{n}$ be the probability that the number of connections required to transmit i and j is equal to n for given $v_1 = i$ and $v_2 = j$, where $n \in \{0, 1, 2\}$ and $1 \leq i, j \leq V$.

For the case of $v_1 \in \mathbb{C}_1$ and $v_2 \in \mathbb{C}_2$, no connection is required to transmit v_1 and v_2. Thus, $\rho^0_{i,j}$ can be defined as

$$\rho^0_{i,j} = \Pr\left[(i \in \mathbb{C}_1) \cap (j \in \mathbb{C}_2)\right], \tag{20}$$

and the average total probability that the number of required connections is 0 can be calculated as

$$\Pr[n=0] = \sum_{i=1}^{V}\sum_{j=1}^{V} f_i f_j \rho^0_{i,j} = \sum_{i=1}^{V}\sum_{j=1}^{V} f_i f_j c_i c_j, \tag{21}$$

by averaging $\rho^0_{i,j}$ over all combinations of i and j.

For given i and j, the probability $\rho^1_{i,j}$ is defined as

$$\rho^1_{i,j} = \Pr\Big[\big[(i \notin \mathbb{C}_1) \cap (j \in \mathbb{C}_2)\big] \cup \big[(i \in \mathbb{C}_1) \cap (j \notin \mathbb{C}_2)\big] \tag{22a}$$

$$\cup \big[(i \notin \mathbb{C}_1) \cap (j \notin \mathbb{C}_2) \cap (i = j)\big] \tag{22b}$$

$$\cup \big[(i \notin \mathbb{C}_1) \cap (j \notin \mathbb{C}_2) \cap (i \neq j) \cap (i \in \mathbb{C}_2) \cap (j \in \mathbb{C}_1)\big]\Big], \tag{22c}$$

where Equation (22a) denote the probability that only one client is supported by UC because only one of v_1 and v_2 is in their own cache. Equation (22b) is the probability that clients are supported by MC because they request the same content, i.e., $v_1 = v_2$, which is not in the clients' caches. Equation (22c) is the probability that clients are supported by XC because each client does not have the own request content, i.e., $v_1 \notin \mathbb{C}_1$ and $v_2 \notin \mathbb{C}_2$, but has the content requested by the other client in cache, $v_1 \in \mathbb{C}_2$ and $v_2 \in \mathbb{C}_1$. In this case, clients 1 and 2 can reconstruct v_i and v_2 from $v_1 \oplus v_2$, respectively. Then, the average total probability that the number of required connections is 1 can be calculated as

$$\begin{aligned}
\Pr[n=1] &= \sum_{i=1}^{V}\sum_{j=1}^{V} f_i f_j \rho^1_{i,j} \\
&= \sum_{i=1}^{V}\sum_{j=1}^{V} f_i f_j \Big[(1-c_i)c_j + c_i(1-c_j)\Big] && \text{(23a)} \\
&+ \sum_{i=1}^{V}\sum_{j=1,j=i}^{V} f_i f_j \Big[(1-c_i)(1-c_j)\Big] && \text{(23b)} \\
&+ \sum_{i=1}^{V}\sum_{j=1,j\neq i}^{V} f_i f_j \Big[(1-c_i)(1-c_j)c_j c_i\Big] && \text{(23c)} \\
&= \sum_{i=1}^{V}\sum_{j=1}^{V} f_i f_j \Big[c_i + c_j - c_i c_j\Big] \\
&+ \sum_{i=1}^{V} f_i^2 \Big[(1-c_i)^2\Big] \\
&+ \sum_{i=1}^{V}\sum_{j=1,j\neq i}^{V} f_i f_j \Big[(1-c_i)(1-c_j)c_j c_i\Big],
\end{aligned}$$

by averaging $\rho^1_{i,j}$ over all combinations of i and j.

Finally, for given i and j, $\rho^2_{i,j}$ is defined as

$$\rho^2_{i,j} = \Pr\left[(i \notin \mathbb{C}_1) \cap (j \notin \mathbb{C}_2) \cap \left[(i \in \mathbb{C}_2) \cap (j \in \mathbb{C}_1)\right]^{\complement}\right], \tag{24}$$

where $A^{\complement}$ denotes the complement of a set A. Then, the average total probability that the number of required connections is 2 can be calculated as

$$\begin{aligned}\Pr[n=2] &= \sum_{i=1}^{V}\sum_{j=1}^{V} f_i f_j \rho_{i,j}^2 \\ &= \sum_{i=1}^{V}\sum_{j=1}^{V} f_i f_j \Big[(1-c_i)(1-c_j)(1-c_j c_i)\Big],\end{aligned} \tag{25}$$

by averaging $\rho_{i,j}^2$ over all combinations of i and j. In this case, clients are supported by UC through two independent connections, respectively.

3.2. A Transmission Algorithm Extended for Multiple Clients ($N \geq 2$)

In this subsection, we propose a transmission algorithm by extending the transmission for two clients described in the previous subsection in order to support XC for N $(N \geq 2)$ clients, while guaranteeing a backward compatibility with conventional streaming schemes such as LC, UC, and MC. Our algorithm consists of two phases. In the first phase, a client n $(1 \leq n \leq N)$ is categorized into LC or MC based on the requesting content, v_n, and cache status, $\mathbb{C}_n$, or proceeds to the second phase. The clients satisfying $v_n \in \mathbb{C}_n$ will be served by LC and become the elements of $\mathbb{U}_L$, which is a set of clients served by LC. The remaining clients with $v_n \notin \mathbb{C}_n$, given by $\mathbb{T} = \{1, \cdots, N\} \setminus \mathbb{U}_L$, transmit a request message to a streaming server including a content to play and cache information. If two or more clients request the same video content, they are all categorized into an MC group. The MC group becomes an element of $\mathbb{G}_M$, which is a set consisting of MC groups, and the cardinality of $\mathbb{G}_M$ is $0 \leq |\mathbb{G}_M| \leq \lfloor \frac{|\mathbb{T}|}{2} \rfloor$ because an MC group consists of at least two clients. $\mathbb{U}_M$ is composed of all clients included in $\mathbb{G}_M$.

In the second phase, the clients who are not included in either $\mathbb{U}_L$ or $\mathbb{U}_M$, given by $\mathbb{T} = \{1, \cdots, N\} \setminus \mathbb{U}_L \setminus \mathbb{U}_M$, are categorized into XC or UC groups. A client n in $\mathbb{T}$ is first checked to see if it can be served by XC. For an existing XC group S, the client n can join the existing XC group S only if the following conditions are satisfied:

$$\begin{aligned}&\mathbb{C}_n \ni v_m \ \forall m \in S, \\ &v_n \in \mathbb{C}_m \ \forall m \in S.\end{aligned} \tag{26}$$

If there is no existing XC group or the client n fails to join any existing XC group, the client n attempts to create a new XC group with another client m who is included in $\mathbb{T}$ and satisfies

$$\begin{aligned}&v_m \in \mathbb{C}_n, \\ &\mathbb{C}_m \ni v_n.\end{aligned} \tag{27}$$

This process is repeated for all clients in $\mathbb{T}$.

For the case that the client n succeeds to join the existing XC group or to create the new XC group, the client n becomes an element of $\mathbb{U}_X$, which is a set consisting the clients in XC groups. Finally, the clients who are not included in either $\mathbb{U}_L$, $\mathbb{U}_M$, or $\mathbb{U}_X$ are included into $\mathbb{U}_U$ and will be served by UC through separate connections. The proposed video streaming scheme using XC is explained in detail in the Algorithms 1 and 2.

Algorithm 1 Proposed Video Streaming:
Phase 1. LC and MC

Initialization:
  $\mathbb{U}_L = \emptyset$, $\mathbb{U}_M = \emptyset$ ▷ Sets of clients
  $\mathbb{G}_M = \emptyset$ ▷ Set of MC groups
Obtain a set of candidate clients for LC:
  $\mathbb{T} = \{1, \cdots, N\}$
for $n \in \mathbb{T}$ **do**
    if $v_n \in \mathbb{C}_n$ **then**
      $\mathbb{U}_L = \mathbb{U}_L \cup \{n\}$
    end if
end for
Obtain a set of candidate clients for MC:
  $\mathbb{T} = \{1, \cdots, N\} \setminus \mathbb{U}_L$
while $|\mathbb{T}| > 0$ **do**
    $u = \mathbb{T}[0]$
    $S = \emptyset$
    for $m \in \mathbb{T}$ **do**
      if $v_u == v_m$ **then**
        $S = S \cup \{m\}$
      end if
    end for
    if $|S| >= 2$ **then**
      $\mathbb{G}_M = \mathbb{G}_M \cup \{S\}$
      $\mathbb{U}_M = \mathbb{U}_M \cup S$
      $\mathbb{T} = \mathbb{T} \setminus S$
    end if
end while

Algorithm 2 Proposed Video Streaming:
Phase 2. XC and UC

Initialization:
$\mathbb{U}_X = \emptyset$ ▷ Set of clients
$\mathbb{G}_X = \emptyset$ ▷ Set of XC groups
Obtain a set of candidate clients for XC:
$\mathbb{T} = \{1, \cdots, N\} \setminus \mathbb{U}_L \setminus \mathbb{U}_M$
$cnt = 0$
while $cnt < |\mathbb{T}|$ and $|\mathbb{T}| > 0$ **do**
    cnt^{++}
    $u = \mathbb{T}[0]$ ▷ A client to check for XC.
    if $|\mathbb{G}_X| > 0$ **then** ▷ XOR group already exists.
        for $S \in \mathbb{G}_X$ **do** ▷ S is a set.
            $j =$ True
            for $s \in S$ **do**
                j *= $(v_u \in \mathbb{C}_s) * (v_s \in \mathbb{C}_u)$
            end for
            if $j ==$ True **then**
                $S = S \cup \{u\}$ ▷ Client u is added into S.
                $\mathbb{U}_X = \mathbb{U}_X \cup \{u\}$
                $\mathbb{T} = \mathbb{T} \setminus \{u\}$
                break
            end if
        end for
    end if
    if $(|\mathbb{G}_X| == 0)$ or $(j ==$ False$)$ **then**
        for $s \in \mathbb{T} \setminus \{u\}$ **do**
            if $(v_u \in \mathbb{C}_s) * (v_s \in \mathbb{C}_u)$ **then**
                $\mathbb{G}_X = \mathbb{G}_X \cup \{\{u, s\}\}$ ▷ A new group of XC
                $\mathbb{T} = \mathbb{T} \setminus \{u, s\}$
                $\mathbb{U}_X = \mathbb{U}_X \cup \{u, s\}$
                break
            end if
        end for
    end if
end while
$\mathbb{U}_U = \{1, \cdots, N\} \setminus \mathbb{U}_L \setminus \mathbb{U}_M \setminus \mathbb{U}_X$

4. Numerical Results

In this section, the performance of video streaming scheme using XC is analyzed in terms of average connections required to serve each client's video play, and compared with that of a conventional scheme without XC. Let κ denote the number of the required connections of the video streaming schemes. For the proposed scheme using XC, κ can be calculated as

$$\kappa_{\text{prop}} = |\mathbb{G}_M| + |\mathbb{G}_X| + |\mathbb{U}_U|, \;\; 0 \leq \kappa_{\text{prop}} \leq N. \tag{28}$$

For the conventional scheme without XC, we need to allocate a dedicated connection to every client in $\mathbb{U}_X$. Thus, κ for the conventional scheme can be calculated as

$$\kappa_{\text{conv}} = |\mathbb{G}_M| + |\mathbb{U}_X| + |\mathbb{U}_U|, \;\; 0 \leq \kappa_{\text{conv}} \leq N. \tag{29}$$

Please note that $\kappa_{\text{prop}} \leq \kappa_{\text{conv}}$ is always satisfied because $|\mathbb{G}_X| \leq |\mathbb{U}_X|$ and $|\mathbb{G}_X| = |\mathbb{U}_X|$ is only valid when $|\mathbb{G}_X| = |\mathbb{U}_X| = 0$. We define the average normalized load as the average number of required connections per each client's video view, which is denoted by η and can be calculated for the proposed and conventional schemes, respectively, as

$$\eta_{\text{prop}} = \sum_{n=0}^{N} \frac{n}{N} \times \Pr[\kappa_{\text{prop}} = n],$$
$$\eta_{\text{conv}} = \sum_{n=0}^{N} \frac{n}{N} \times \Pr[\kappa_{\text{conv}} = n]. \quad (30)$$

All numerical results were obtained through mathematical analysis given in Section 3.1 and verified by Monte-Carlo simulations using a Python-based simulator. In the figures, the lines and markers denote numerical results from mathematical analysis and Monte-Carlo simulations, respectively. In simulations, the number of iterations for averaging the effect of many random variables is 100,000.

Figure 5 shows the average normalized load η of the proposed streaming scheme using XC and the conventional scheme without XC for various values of cache length. The numbers of clients and video contents are given by $N = 2$ and $V = 20$, respectively, and the parameter α of Zipf distribution for contents' normalized popularity is 0, 0.5, 1.0, or 1.5. The figure shows that our numerical results obtained by mathematical analysis are consistent with those from simulations. The proposed and conventional schemes can both reduce the average load as the parameter α of Zipf distribution increases. As α increases, the deviation of popularity among contents is expanded and thus every client is more likely to play the popular contents, which indicates that the probabilities that the popular contents are cached increase while the probabilities that less popular contents are cached decrease. As a result, an increasing α can reduce the average normalized load because the hit probability, which is the probability that the contents to play are cached in their own caches, increases. As the length of cache, C, increases, the probabilities that contents are cached increase regardless of popularity. Thus, for given α, an increasing C also reduces the average normalized load. To be more specific, when $\alpha = 1.5$ and C increases from 1 to 5, the average normalized load decreases from 0.7 to 0.3 for both proposed and conventional schemes. In this figure, although the proposed scheme using XC outperforms the conventional scheme without XC, the gap of the performance is marginal, because when the number of clients is small $(N = 2)$, the frequency of using XC decreases and hence, the gain achieved by XC is limited.

In Figure 6, we compare the performance of the proposed scheme to the conventional scheme according to the number of clients, N. Figure 6a shows the average normalized load of the proposed and conventional schemes by varying N from 2 to 10 when $V = 20$, $C \in \{1, 2, 3, 4, 5\}$ and $\alpha = 0$. Here, $\alpha = 0$ means that Zipf distribution become Uniform distribution and all video contents thus have the same popularity. As N increases, the frequency of using MC for the conventional scheme and the frequency of using MC or XC for the proposed scheme both increase. Thus, the average normalized load decreases with an increasing N for both the proposed and conventional schemes, as predicted in Figure 5. In addition, the average load also decreases with an increasing C because the frequency of using LC for the conventional scheme and the frequency of using LC or XC for the proposed scheme both increase as C increases. Figure 6b shows how much the proposed scheme can reduce the average normalized load by using XC, compared to the conventional scheme. The percentage of average load reduction by the proposed scheme is calculated as $\zeta = \frac{\eta_{\text{conv}} - \eta_{\text{prov}}}{\eta_{\text{conv}}} \times 100[\%]$. In this figure, we can observe that for all N and C values, the proposed scheme outperforms the conventional scheme due to the gain obtained from XC, and thus $\zeta > 0$, $\forall N$, C. The frequency of using XC in the proposed scheme increases as N or C increases and thus, ζ increases with increasing N or C. As a specific example with $C = 5$ and $N = 10$, the proposed scheme can reduce the average load about 18%, compared to the conventional scheme.

In Figure 7, according to N, we plot the average normalized load in Figure 7a and the percentage of average load reduction, ζ, in Figure 7b, respectively, when $V = 20$ and $C \in \{1, 2, 3, 4, 5\}$. In this figure, α is set to 1.0 to let all videos have different popularity from each other. For the case of $\alpha = 1.0$, the frequency of viewing top three out of 20 videos accounts for about 50%. The average normalized load is reduced more for both proposed and conventional schemes as α increases from 0 to 1.0, as shown in Figures 6a and 7a. As α increases, the deviation in popularity among video contents increases, which can yield the same effect as increasing N or C. By comparing Figures 6a and 7a, we can observe that as α increases from 0 to 1.0, the average load of the proposed scheme decreases from 0.523 to 0.39 when $C = 5$ and $N = 10$. In terms of the percentage of average load reduction, by comparing Figures 6b and 7b, we can observe that ζ is determined by C and N as well as α. For the case of $N = 10$, when C is relatively large, e.g., $C = 5$, ζ decreases from 17.8% to 15.3% as α increases from 0 to 1.0. In this case, however, when $C = 1$, 2, or 3, ζ with $\alpha = 1.0$ is greater than ζ with $\alpha = 0$ for most N values. For example, when $C = 3$ and $N = 5$, ζ increases from 5.5% to 7.3% as α increases from 0 to 1.0. In the proposed scheme, LC and MC are given a higher priority than XC. Therefore, as C, N, and α all increase, the probability of using LC or MC increases and the gain achieved from XC decreases slightly. Even though we used small values for N, V, and C to simplify computer simulations, the proposed scheme can be applied to real large-scaled video stream systems.

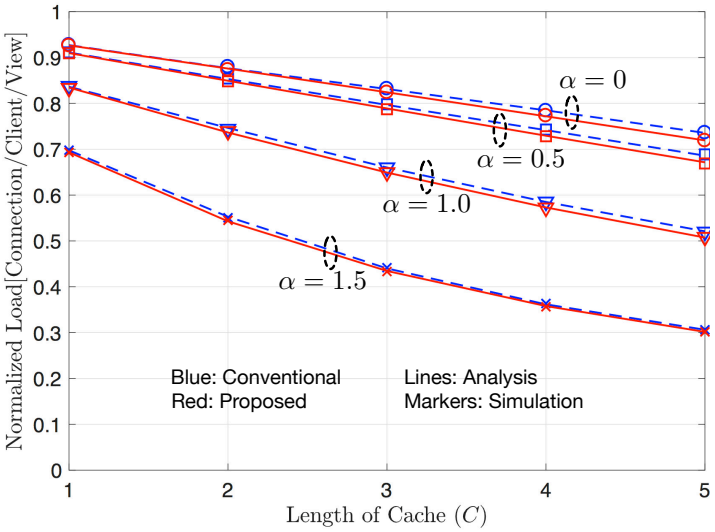
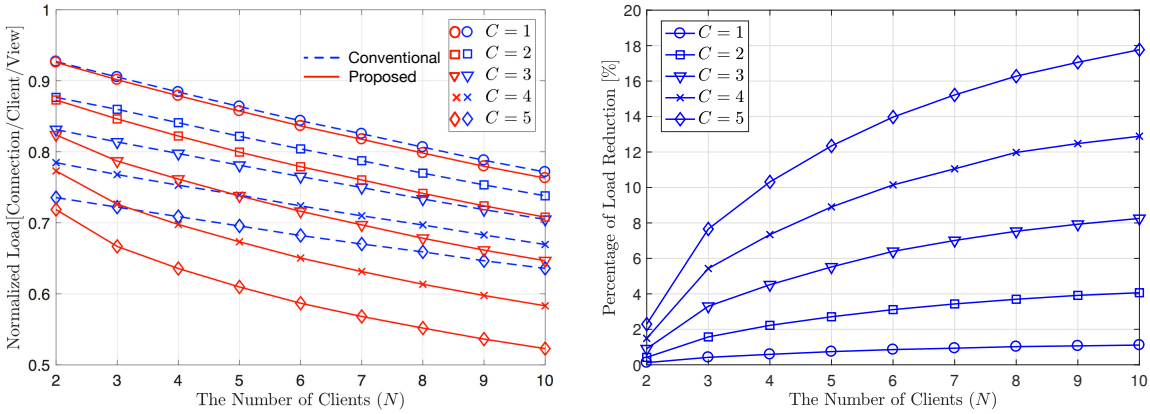
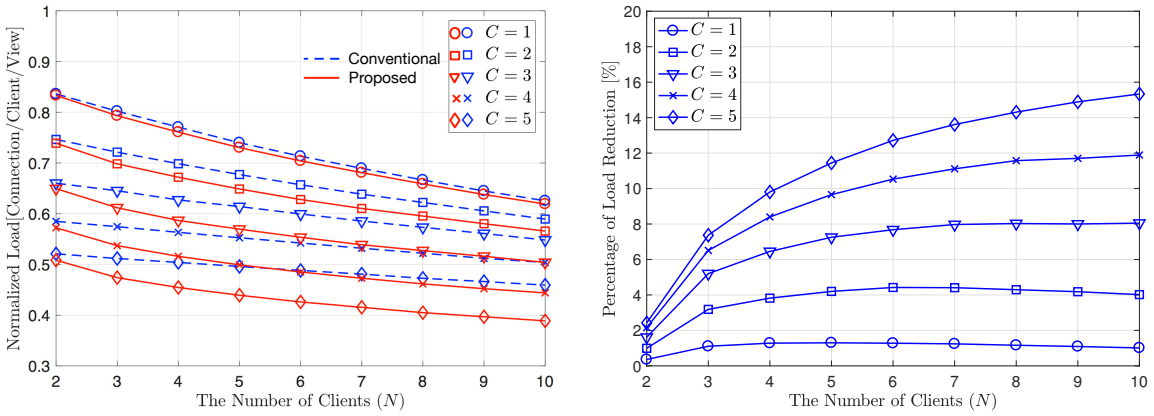


Figure 5. Average normalized load for various values of C. N=2, $V = 20$, and $\alpha \in \{0, 0.5, 1.0, 1.5\}$.



(**a**) Average normalized load. (**b**) Percentage of load reduction of the proposed scheme.

Figure 6. Average normalized load and percentage of load reduction of the proposed scheme for varying N. $V = 20$, $C \in \{1, 2, 3, 4, 5\}$, and $\alpha = 0$.



(**a**) Average normalized load. (**b**) Percentage of load reduction of the proposed scheme.

Figure 7. Average normalized load and percentage of load reduction of the proposed scheme for varying N. $V = 20$, $C \in \{1, 2, 3, 4, 5\}$, and $\alpha = 1.0$.

5. Conclusions

A streaming server can process requests from multiple clients simultaneously by coding the bit streams requested from the multiple clients with XOR operation and multicasting the coded bit stream to the multiple clients. Each client can recover its content from the coded bit stream with XOR operation using contents stored in its cache, which is based on zero-identity and self-inverse properties of XOR and can reduce the number of transmissions by dealing with the requests simultaneously. Existing studies on the data transmission using XOR-based coding mainly focused on theoretical analysis or optimal coding in limited environments, and most of them were thus less feasible. In this paper, we investigated how the data transmission using XOR-based coding could be applied to video streaming systems without interfering existing video streaming methods such as UC, MC, and BC. We first modeled the behavior of video contents in clients' caches as a Markov chain, and derived the caching probability that a video content has been already cached in the client's cache when a client wants to play the content at an arbitrary moment. We also proposed a new video streaming scheme for supporting XC while guaranteeing a backward compatibility with the conventional video streaming methods. With a simplified case of two clients, the basic concept of the proposed scheme was described, and the performance of the proposed scheme mathematically analyzed in terms of average normalized load. All analytical results were verified by computer simulations. Then, the proposed scheme was extended to a complete algorithm that can support multiple clients more than two. Our numerical results showed that the proposed scheme using XC outperforms the conventional scheme without XC in all situations and the performance enhancement of the proposed scheme becomes more noticeable as the number of clients or the length of cache increase. For a specific situation with $C = 5$ and $N = 10$, we showed the proposed scheme can reduce the average load about 18%, compared to the conventional scheme.

Author Contributions: Conceptualization and problem formulation, T.-W.B.; writing—original draft preparation, T.-W.B.; methodology and formal analysis, W.L.; visualization and simulations, W.L.; funding acquisition, T.-W.B. writing—review and editing, J.R. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(Ministry of Education) (No. 2020R1I1A3061195, Development Of Wired and Wireless Integrated Multimedia-Streaming System Using Exclusive OR-based Coding).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cisco White Paper. *Cisco Visual Networking Index: Forecast and Trends, 2017–2022*. 2018. Available online: https://cloud.report/whitepapers/cisco-visual-networking-index-forecast-and-trends-2017-2022/9017 (accessed on 31 October 2020).
2. Yang, P.; Zhang, N.; Zhang, S.; Yu, L.; Zhang, J.; Shen, X. Content popularity prediction towards location-aware mobile edge caching. *IEEE Trans. Multimed.* **2019**, *21*, 915–929. [CrossRef]
3. Nikolaou, S.; Renesse, R.; Schiper, N. Proactive cache placement on cooperative client caches for online social networks. *IEEE Trans. Parallel Distrib. Syst.* **2016**, *27*, 1174–1186. [CrossRef]
4. Vakali, A.; Pallis, G. Content delivery networks: Status and trends. *IEEE Internet Comput.* **2003**, *7*, 68–74. [CrossRef]
5. Kusmierek, E.; Dong, Y.; Du, D.H.C. Loopback: Exploiting collaborative caches for large-scale streaming. *IEEE Trans. Multimed.* **2006**, *8*, 233–242. [CrossRef]
6. Sun, G.; Al-Ward, H.; Boateng, G.O.; Liu, G. Autonomous cache resource slicing and content placement at virtualized mobile edge network. *IEEE Access* **2019**, *7*, 84727–84743. [CrossRef]
7. Chae, S.H.; Choi, W.; Caching placement in stochastic wireless caching helper networks: Channel selection diversity via caching. *IEEE Trans. Wirel. Commun.* **2016**, *15*, 6626–6637. [CrossRef]
8. Chae, S.H.; Quek, T.Q.S.; Choi, W. Content placement for wireless cooperative caching helpers: A tradeoff between cooperative gain and content diversity gain. *IEEE Trans. Wirel. Commun.* **2017**, *16*, 6795–6807. [CrossRef]
9. Zou, J.; Li, C.; Zhai, C.; Xiong, H.; Steinbach, E. Joint pricing and cache placement for video caching: A game theoretic approach. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 1566–1583. [CrossRef]
10. Jeon, S.; Hong, S.; Ji, M.; Caire, G.; Molisch, A.F. Wireless multihop device-to-device caching networks. *IEEE Trans. Inf. Theory* **2017**, *63*, 1662–1676. [CrossRef]
11. Zhang, X.; Zhu, Q. Distributed mobile devices caching over edge computing wireless networks. In Proceedings of the 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Atlanta, GA, USA, 1–4 May 2017; pp. 127–132.
12. Liu, D.; Yang, C. A deep reinforcement learning approach to proactive content pushing and recommendation for mobile users. *IEEE Access* **2019**, *7*, 83120–83136. [CrossRef]
13. Tao, M.; Gündüz, D.; Xu, F.; Roig, J.S.P. Content caching and delivery in wireless radio access networks. *IEEE Trans. Commun.* **2019**, *67*, 4724–4749. [CrossRef]
14. Bidokhti, S.S.; Wigger, M.; Timo, R. Noisy broadcast networks with receiver caching. *IEEE Trans. Inf. Theory* **2018**, *64* 6996–7016. [CrossRef]
15. Yang, C.; Xia, B.; Xie, W.; Huang, K.; Yao, Y.; Zhao, Y.; "Interference cancelation at receivers in cache-enabled wireless networks. *IEEE Trans. Veh. Technol.* **2018**, *67*, 842–846. [CrossRef]
16. Birk, Y.; Kol, T. Coding on demand by an informed source (iscod) for efficient broadcast of different supplemental data to caching clients. *IEEE Trans. Inf. Theory* **2006**, *52*, 2825–2830. [CrossRef]
17. Blasiak, A.; Kleinberg, R.; Lubetzky, E. Index coding via linear programming. *arXiv* **2010**, arXiv:1004.1379.
18. Arbabjolfaei, F.; Bandemer, B.; Kim, Y.; Şaşoğlu, E.; Wang, L. On the capacity region for index coding. In Proceedings of the 2013 IEEE International Symposium on Information Theory (ISIT), Istanbul, Turkey, 7–12 July 2013; pp. 962–966.
19. Son, K.; Lee, J.H.; Choi, W. User-cache aided transmission with index coding in K -user downlink channels. *IEEE Trans. Wirel. Commun.* **2019**, *18*, 6043–6058. [CrossRef]
20. Hefeeda, M.; Saleh, O. Traffic modeling and proportional partial caching for peer-to-peer systems. *IEEE/ACM Trans. Netw.* **2008**, *16*, 1447–1460. [CrossRef]
21. Cha, M.; Kwak, H.; Rodriguez, P.; Ahn, Y.; Moon, S. Analyzing the video popularity characteristics of large-scale user generated content systems. *IEEE/ACM Trans. Netw.* **2009**, *17*, 1357–1370.
22. Eriksson, M.; Rahman, S.M.H.; Fraile, F.; Sjöström, M. Efficient interactive multicast over DVB-T2—Utilizing dynamic SFNs and PARPS. In Proceedings of the 2013 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), London, UK, 5–7 June 2013; pp. 1–7.

23. Zhong, C.; Gursoy, M.C.; Velipasalar, S. Deep reinforcement learning-based edge caching in wireless networks. *IEEE Trans. Cogn. Commun. Netw.* **2020**, *6*, 48–61. [CrossRef]
24. Grinstead, C.M.; Snell, J.L. *Introduction to Probability*; American Mathematical Soc.: Providence, RI, USA, 2012.

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).