sensors

MDPI

Article

A New Cache Update Scheme Using Reinforcement Learning for Coded Video Streaming Systems

Yu-Sin Kim [1], Jeong-Min Lee [2], Jong-Yeol Ryu [2] and Tae-Won Ban [2,*]

[1] Algorithm Team, Carvi, Seoul 08513, Korea; usin1216@carvi.co.kr
[2] Department of Information and Communication Engineering, Gyeongsang National University, Gyeongnam 53064, Korea; ljmin200002@gmail.com (J.-M.L.); jongyeol_ryu@gnu.ac.kr (J.-Y.R.)
* Correspondence: twban35@gnu.ac.kr

Abstract: As the demand for video streaming has been rapidly increasing recently, new technologies for improving the efficiency of video streaming have attracted much attention. In this paper, we thus investigate how to improve the efficiency of video streaming by using clients' cache storage considering exclusive OR (XOR) coding-based video streaming where multiple different video contents can be simultaneously transmitted in one transmission as long as prerequisite conditions are satisfied, and the efficiency of video streaming can be thus significantly enhanced. We also propose a new cache update scheme using reinforcement learning. The proposed scheme uses a *K*-actor-critic (*K*-AC) network that can mitigate the disadvantage of actor-critic networks by yielding *K* candidate outputs and by selecting the final output with the highest value out of the *K* candidates. The *K*-AC exists in each client, and each client can train it by using only locally available information without any feedback or signaling so that the proposed cache update scheme is a completely decentralized scheme. The performance of the proposed cache update scheme was analyzed in terms of the average number of transmissions for XOR coding-based video streaming and was compared to that of conventional cache update schemes. Our numerical results show that the proposed cache update scheme can reduce the number of transmissions up to 24% when the number of videos is 100, the number of clients is 50, and the cache size is 5.

Keywords: streaming; multimedia; reinforcement learning; cache; exclusive OR

Citation: Kim, Y.-S.; Lee, J.-M.; Ryu, J.-Y.; Ban, T.-W. A New Cache Update Scheme Using Reinforcement Learning for Coded Video Streaming Systems. *Sensors* **2021**, *21*, 2867. https://doi.org/10.3390/s21082867

Academic Editor: Nikolaos Thomos

Received: 10 March 2021
Accepted: 15 April 2021
Published: 19 April 2021

1. Introduction

In recent years, Internet traffic has been rapidly increasing and is expected to increase more rapidly in the future [1,2]. In particular, it is also expected that video streaming traffic will account for 82% of the global Internet traffic by 2022 due to the wide popularity of various video streaming platforms such as YouTube [1]. This trend is more pronounced in mobile networks, and many advanced techniques have been thus investigated to increase the capacity of next-generation mobile communication networks [3–5]. Along with many technologies to increase network capacity by using a wide bandwidth or by increasing spectral efficiency, other technologies for reducing network traffic are also attracting much attention as another alternative [6,7]. Multicast (MC) transmission can reduce network traffic by transmitting a video to multiple clients in one transmission if the clients requested the same video at the same time [6]. Proxy servers with cache can significantly reduce network traffic, and bandwidth optimization for real-time video traffic transmission through a proxy server was investigated in [7]. In particular, MC-aware caching can better exploit the available cache space and can yield a gain of 19% over existing caching schemes [6]. Many studies have studied how to reduce network traffic by using the transmitters' cache storage, while the low cost and large capacity of storage motivated some studies to focus on the clients' cache storage [8–13]. In this paper, we thus investigate a new video streaming system using clients' cache and XOR-based index coding. In the new video streaming

system, multiple different video contents can be transmitted in one transmission if prerequisites are satisfied, and transmission efficiency can be thus significantly improved. Cache update is an important factor in video streaming systems [14–19]. However, there have been no previous studies that investigated cache update policies for the index coding-based video streaming system. Thus, we investigate how to update the clients' cache for index coding-based video streaming systems in order to use the clients' cache more efficiently, and we propose a new cache update scheme for clients using deep reinforcement learning. The proposed cache update scheme was based on a new architecture called *K*-actor-critic (*K*-AC) that can mitigate the shortcomings of the actor-critic (AC) network architecture. The *K*-AC network that consists of an actor network and the main value network exists in each client, and each client can thus update its own cache in a fully decentralized manner without any exchange of information or signaling. In this work, we assumed that all clients have different popularity for videos, and the popularity for each client is time varying, contrary to most conventional studies assuming that video popularity is the same for all clients and is time invariant.

The rest of this paper is organized as follows. We investigate related studies in Section 2. Section 3 introduces the system model considered in this paper and describes the basic concept of XOR coding-based video streaming. A mathematical ground for reducing the number of XOR operations is also introduced in Section 3. In Section 4, we propose a new cache update algorithm using reinforcement learning for index coding-based video streaming systems. Section 5 shows the numerical results. Finally, this paper is concluded in Section 6.

2. Related Work

Contrary to conventional strategies that used the transmitters' cache, there have been recent studies to exploit the clients' cache storage [8–13]. Methods that can efficiently exploit the clients' cache storage were investigated from the viewpoint of information theory [8–10]. Lower and upper bounds were presented on the capacity-memory tradeoff of an erasure broadcast network with two disjoint sets of receivers: a set of weak receivers with equal erasure probabilities and equal cache sizes and a set of strong receivers with equal erasure probabilities and no cache memories [8]. It was proposed to exploit the limited cache packets as side information to cancel incoming interference at the receiver side by considering a stochastic network [9]. A new inner bound on the capacity region of the general index coding problem was investigated by relying on a random coding scheme and optimal decoding [10]. A new concept using index coding for transmitting contents was proposed in [11], where multiple contents were index coded, and they can be transmitted in one transmission over a single channel if some prerequisites are satisfied. A new algorithm of the index code and time resource allocation that can minimize wireless transmission outage probability with a low complexity was proposed [12]. Many studies focusing on the clients' cache mainly investigated theoretical performance analysis or optimal index code design by considering simplistic or unrealistic system models, while the index code was applied to a realistic system in [13]. Exclusive OR (XOR)-based index coding can be applied to large-scale video streaming systems while providing a complete backward compatibility with existing streaming schemes such as unicast (UC) and MC thanks to the properties of the XOR operator such as zero-identity, self-inverse, commutativity, and associativity [13].

On the other hand, there have been many studies on cache update [14–19]. The performance of FIFO, the least recently used (LRU), and the least frequently used (LFU) schemes was analyzed in terms of the rate at which a particular request is returned before a given deadline [14] and in terms of hit rate [15]. A novel content-aware cache replacement algorithm taking advantage of content demand forecasts was investigated to efficiently use limited caches in size [16]. LRU-K, which is a combination of LRU and LFU, was proposed [17]. They simulated TV distribution with time-shift and investigated the effect of introducing a local cache close to the viewers and what impact TV program popularity, program set size, cache replacement policy, and other factors had on the caching efficiency [18].

A new concept that cache servers share request information to predict the popularity of contents through regression was proposed [19]. A deep Q-network (DQN)-based cache update scheme for edge cache networks was proposed [20]. They aimed at maximizing the overall quality of 360° videos delivered to the end-users by caching the most popular ones at base quality along with a virtual viewport in high quality. A new centralized cache update scheme using the Wolpertinger architecture for base stations was proposed [21]. The Wolpertinger architecture selects a single proto-action from the actor network and selects the K-closest action around the proto-action for the input of the critic network [22]. Contrary to the Wolpertinger architecture, our K-AC directly selects K candidate actions with the highest Q values from the actor network for the input of the critic network, inspired by the fact that the actions in our problem do not have a strong correlation with each other. Despite these many existing studies on cache update, the simplest cache update scheme, first-in first-out (FIFO), was only considered in index coding-based video streaming systems [13], and there have been no cache update schemes targeting index coding-based video streaming systems. In index coding-based video streaming systems, each client needs to update its cache so as to increase the probability of index coding with other clients, as well as its own hit probability, contrary to conventional video streaming systems where each client's hit probability is only considered.

3. XOR Coding-Based Streaming System

We investigated a coded video streaming system, as depicted in Figure 1, which consisted of N clients and a streaming server. All the clients and the server were equipped with cache. Clients' cache can store C videos, while the server's cache can store V videos ($V \gg C$). It was assumed that all videos had the same length in time. Even if multiple clients request different videos, they can be selectively XOR-encoded into one bit stream according to the status of their caches [13]. For a given set of clients, if every client in the set has all videos requested by the remaining clients in its cache, then all the clients in the set can receive their videos through XOR coding in one transmission. This is called XOR-cast (XC). The XOR-encoded bit stream is transmitted to the clients by one transmission, and we can reduce the number of transmissions for the videos requested by the clients. Then, each client restores its video by decoding the received bit stream with the contents stored in its cache [13]. As a specific example, the client requesting v_1 in Figure 1 plays the video v_1 stored in its cache without receiving any data from the server, which is called local cast (LC). The two clients requesting v_2 can stream v_2 from the same channel through MC. The client requesting v_3 and the client requesting v_4 store v_4 and v_3, respectively, and the server thus XOR encodes v_3 and v_4. $(v_3 \oplus v_4)$ is transmitted over a single channel through XC even though v_3 and v_4 are different. The client that requested v_3 restores v_3 by using $(v_3 \oplus v_4) \oplus v_4 = v_3$, and the client that requested v_4 restores v_4 by using $(v_3 \oplus v_4) \oplus v_3 = v_4$, where the equalities are valid due to the properties of the XOR operator such as zero-identity, self-inverse, commutativity, and associativity.

The relative popularity of the v-th most popular one among V videos is modeled by the Zipf distribution, which is given by:

$$f(v;\beta,V) = \frac{1/v^{\beta}}{\sum_{k=1}^{V}(1/k^{\beta})}, \tag{1}$$

where β is the Zipf parameter characterizing the distribution and $\sum_{v=1}^{V} f(v;\beta,V) = 1$ regardless of β [23]. Contrary to most conventional studies that assumed that all clients have the same relative popularity for all videos and the relative popularity is time-invariant, we assumed that all clients have different popularity and that the popularity for each client

is time varying. Client n requests a video v at time t with a probability $P^t_{(n,v)}$. $P^t_{(n,v)}$'s are time varying and different for all clients and can be defined as:

$$P^{t+1}_{(n,v)} = \begin{cases} \rho P^t_{(n,v)} + (1-\rho) f(w;\beta,V) & \text{with prob. } p \\ P^t_{(n,v)} & \text{with prob. } 1-p, \end{cases} \tag{2}$$

where p denotes the probability that the rank v of a video changes to a new rank w for the client n, w denotes that the new rank of the video v is a random integer between one and V, and ρ denotes a correlation between the old rank v and the new rank w satisfying $0 < \rho < 1$ for all $v \in \{1, \ldots, V\}$. The initial probability of $P^t_{(n,v)}$ is given by $P^0_{(n,v)} = f(v;\beta,V)$. p and ρ can adjust the frequency and the amount of change in popularity for video v, respectively.

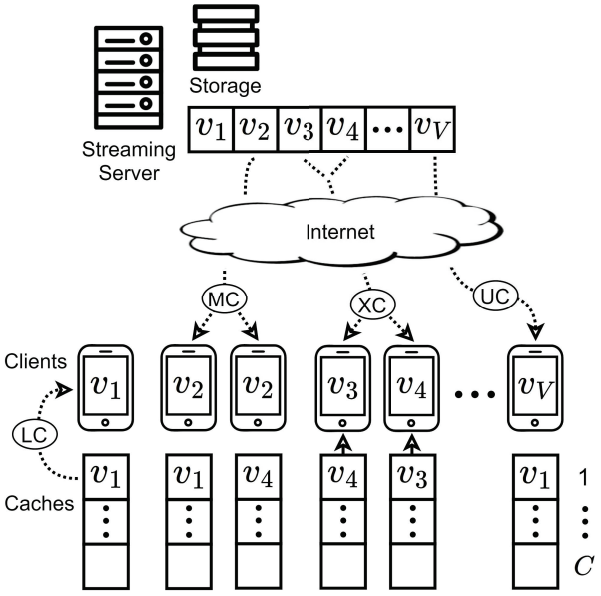

Figure 1. System architecture.

Figure 2 shows the overall procedure of XOR coding-based streaming systems. r_n and $\mathcal{C}_n$ denote a video that client n requests and the set of videos stored in the cache of the client n, respectively. $|\mathcal{C}_n| = C$, where $|\dot{|}$ denotes the cardinality of a set. In this system, we aimed to reduce the number of transmissions required to transmit the N videos $\{r_n | n \in \mathcal{U}\}$ requested by the N clients, where $\mathcal{U}$ denotes the set of the whole clients and is given by $\mathcal{U} = \{1, 2, \ldots, N\}$. If $r_n \in \mathcal{C}_n$, which denotes that r_n is stored in the client n's cache, then the client n can play the r_n stored in the cache through LC without connecting to the server. The set of clients who can play a video through LC can be found as:

$$\mathcal{G}_{\text{LC}} = \{n | r_n \in \mathcal{C}_n,\ n \in \mathcal{U}\}. \tag{3}$$

If an arbitrary client n is not included in $\mathcal{G}_{\text{LC}}$, it transmits a request message including the information of r_n and $\mathcal{C}_n$ to the server. The extra overhead per client required to send $\mathcal{C}_n$, denoted by $\mathcal{O}$, can be calculated as:

$$\mathcal{O} = \lceil \log_2 V \rceil \times C, \tag{4}$$

where $\lceil \cdot \rceil$ denotes the ceiling function. $\mathcal{O}$ is linearly proportional to C, which is not a big value in real environments and is logarithmically proportional to V. In addition, $\mathcal{O}$ is ignorable, compared to the size of recent video contents. If there exist multiple clients

that have requested the same video, they can all receive the video through MC in one transmission. The set of clients who can receive a video through MC can be found as:

$$\mathcal{G}_{\mathrm{MC}} = \left\{ n \;\middle|\; \left| \left\{ i | r_i = r_n, (i \neq n) \& (i \in \mathcal{U} \setminus \mathcal{G}_{\mathrm{LC}}) \right\} \right| \geq 1, n \in \mathcal{U} \setminus \mathcal{G}_{\mathrm{LC}} \right\}, \tag{5}$$

where $A \setminus B$ denotes the set difference of sets A and B. $\mathcal{G}_{\mathrm{MC}}$ includes all clients that can receive a video through MC, and the number of transmissions required for $\mathcal{G}_{\mathrm{MC}}$ denoted by K_{MC} can be calculated by:

$$K_{\mathrm{MC}} = \left| \sum_{n \in \mathcal{G}_{\mathrm{MC}}} \{r_n\} \right|, \tag{6}$$

where $(\mathcal{A} + \mathcal{B})$ denotes the union of two sets $\mathcal{A}$ and $\mathcal{B}$, removing duplicate elements instead of the arithmetic addition for notational simplicity. Then, all remaining clients that are not included in $\mathcal{G}_{\mathrm{LC}}$ or $\mathcal{G}_{\mathrm{MC}}$, given by $\mathcal{X} = \mathcal{U} \setminus \mathcal{G}_{\mathrm{LC}} \setminus \mathcal{G}_{\mathrm{MC}}$, become candidates for XC, and the server sorts out the clients eligible for XC. A client $i \in \mathcal{X}$ can receive a video content through XC together with other clients in $\mathcal{X}$ that satisfy $\{j | r_i \in \mathcal{C}_j, r_j \in \mathcal{C}_i, j \neq i, j \in \mathcal{X}\}$. They compose one group for XC, and the server XOR encodes their video contents into one bit stream and transmits the bit stream in one transmission. For each client i in $\mathcal{X}$, the server looks for other clients in $\mathcal{X}$ that can be grouped with the client i for XC, and the result can be obtained by:

$$\mathbb{G}_{\mathrm{XC}} = \{ \{i\} + \{j | r_i \in \mathcal{C}_j, r_j \in \mathcal{C}_i, j \neq i, j \in \mathcal{X}\} | i \in \mathcal{X} \}. \tag{7}$$

$\mathbb{G}_{\mathrm{XC}}$ is a set of sets and $\mathbb{G}_{\mathrm{XC}}[k]$ denotes the k-th element of $\mathbb{G}_{\mathrm{XC}}$, which is a set. If $\mathbb{G}_{\mathrm{XC}}[k]$ includes a single client, $|\mathbb{G}_{\mathrm{XC}}[k]| = 1$, the client will receive the video by UC, and if $|\mathbb{G}_{\mathrm{XC}}[k]| = 2$, the two clients will receive their videos by XC with no other options. If $|\mathbb{G}_{\mathrm{XC}}[k]| \geq 3$, the possibility of XC among the rest of the clients except for $\mathbb{G}_{\mathrm{XC}}[k][1]$ exists, and there can be thus multiple options that the clients can be grouped for XC. We need to reduce the number of XOR operations, and the number of XOR operations decreases as the cardinalities of XC groups are even, as described in Theorem 1 and Remark 1. The server sorts all groups in $\mathbb{G}_{\mathrm{XC}}$ in ascending order according to their cardinalities and saves them in $\hat{\mathbb{G}}_{\mathrm{XC}}$. $\hat{\mathbb{G}}_{\mathrm{XC}}[\hat{k}]$ denotes the group with the k-th smallest cardinality, and $|\hat{\mathbb{G}}_{\mathrm{XC}}[\hat{k}]| \leq |\hat{\mathbb{G}}_{\mathrm{XC}}[\widehat{k+1}]|$ is thus satisfied for all k's, $1 \leq k \leq |\mathcal{X}| - 1$. Then, XC groups can be obtained by:

$$\tilde{\mathbb{G}}_{\mathrm{XC}}[i] = \hat{\mathbb{G}}_{\mathrm{XC}}[i] \setminus \sum_{j=1}^{i-1} \hat{\mathbb{G}}_{\mathrm{XC}}[j], \tag{8}$$

where all duplicate groups are removed and smaller XC groups are chosen instead of larger ones when there are multiple options for XC grouping. Finally, the set of clients who can stream a video through XC can be given as:

$$\mathcal{G}_{\mathrm{XC}} = \sum_{i=1}^{|\tilde{\mathbb{G}}_{\mathrm{XC}}|} \tilde{\mathbb{G}}_{\mathrm{XC}}[i], \tag{9}$$

and the number of transmissions required for $\mathcal{G}_{\mathrm{XC}}$ is denoted by K_{XC} and can be calculated by:

$$K_{\mathrm{XC}} = |\tilde{\mathbb{G}}_{\mathrm{XC}}|. \tag{10}$$

As a specific example, assume that $\mathbb{G}_{\mathrm{XC}} = \{\{1,2,3\}, \{2,1,3\}, \{3,1,2,4\}, \{4,3\}\}$. Then, two different options for making XC groups exist; $A : \{\{1,2\}, \{3,4\}\}$, which requires

six XOR operations, and $B : \{\{1,2,3\},\{4\}\}$, which requires eight XOR operations. Even though the two options both require two transmissions, $\hat{\mathbb{G}}_{\mathrm{XC}}$ is given as:

$$\hat{\mathbb{G}}_{\mathrm{XC}} = \{\{4,3\},\{1,2,3\},\{2,1,3\},\{3,1,2,4\}\}, \tag{11}$$

and $\tilde{\mathbb{G}}_{\mathrm{XC}}$ is calculated as:

$$\tilde{\mathbb{G}}_{\mathrm{XC}} = \{\{4,3\},\{1,2\}\} \tag{12}$$

by (8). Thus, the option A with two XC groups $\{1,2\}$ and $\{3,4\}$ is chosen instead of the option B by (8) due to its smaller number of XOR operations, where six is the minimum number of XOR operations for $K = 2$ and $N = 4$, given by Theorem 1. $\mathcal{G}_{\mathrm{XC}} = \{1,2,3,4\}$.

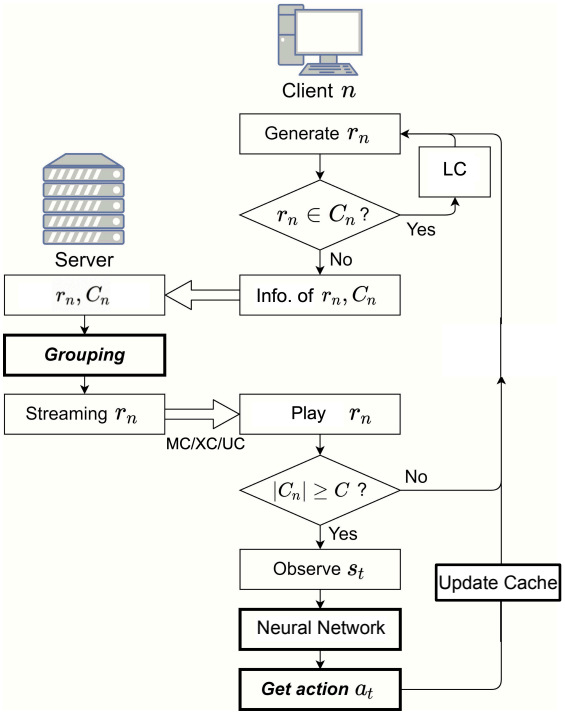

Figure 2. Overall procedures of XOR coding-based streaming.

Theorem 1. *For M XC groups with N clients, the minimum total number of XOR operations required by the server and the clients is* $\frac{N^2}{M} - M$.

Proof. For an XC group consisting of n clients, the server requires $(n-1)$ XOR operations for encoding, and each client in the XC group also requires $(n-1)$ XOR operations for decoding. Thus, the total number of XOR operations required by the server and the clients can be calculated by $(n-1) + n(n-1) = n^2 - 1$. If we have M XC groups and N clients in total and N_k denotes the cardinality of the i-th XC group, the total number of XOR operations required by both the server and the clients can be calculated as:

$$\mathsf{O} = \sum_{i=1}^{M} (N_i^2 - 1) = \sum_{i=1}^{M} N_i^2 - M, \tag{13}$$

where $\sum_{i=1}^{M} N_i^2$ can be rewritten as $\sum_{i=1}^{M} N_i^2 = M\frac{\sum_{i=1}^{M} N_i^2}{M} = ME[N_i^2]$. For an arbitrary random variable X, $V[X] = E[X^2] - E[X]^2$. Thus, (13) can be rewritten as:

$$\begin{aligned} \mathsf{O} &= M\Big(E[N_i^2] - 1\Big) \\ &= M\Big(E[N_i]^2 + V[N_i] - 1\Big) \\ &= M\left(\left(\frac{\sum_{i=1}^{M} N_i}{M}\right)^2 + V[N_i] - 1\right) \\ &= \frac{(\sum_{i=1}^{M} N_i)^2}{M} - M + MV[N_i] \\ &= \frac{N^2}{M} - M + MV[N_i], \end{aligned} \tag{14}$$

where the third equality is valid because $E[N_i] = \frac{\sum_{i=1}^{M} N_i}{M}$. The minimum value of O is $\frac{N^2}{M} - M$, which is achieved when $V[N_i] = 0$ because $V[N_i]$ is non-negative. This completes the proof of Theorem 1. □

Remark 1. *For M XC groups with N clients in total, the total number of XOR operations decreases as the variance of the cardinalities of XC groups decreases.*

In this paper, we placed a higher priority on MC over XC to reduce the computational complexity for XC grouping and XOR coding by decreasing the number of candidate clients of XC without increasing the number of required transmissions. Finally, all the remaining clients, given by $\mathcal{G}_{\mathrm{UC}} = \mathcal{U} \setminus \mathcal{G}_{\mathrm{LC}} \setminus \mathcal{G}_{\mathrm{MC}} \setminus \mathcal{G}_{\mathrm{XC}}$, will receive their videos through UC. The number of transmissions required for UC is calculated by $K_{\mathrm{UC}} = |\mathcal{G}_{\mathrm{UC}}|$.

4. Proposed Cache Update Scheme Using Reinforcement Learning

In this section, we formulate a cache management problem for XOR coding-based streaming systems and propose a new cache update scheme using reinforcement learning to improve the efficiency of video streaming. In our problem, each client updates its cache by replacing a content stored in $\mathcal{C}_n$ with r_n after playing r_n.

In conventional actor-critic (AC) networks, one action is only generated by actor networks, and the action may not be thus optimal with a high probability; it is also difficult to evaluate the value of the action generated by the actor network. In this paper, we thus proposed the *K*-actor-critic (*K*-AC) network to overcome the disadvantage of AC networks, which is depicted in Figure 3. The *K*-AC exists in each and every client and consists of an actor network and the main value network. s_t and $\pi(s_t)$ denote the input state and the output of actor network, respectively. s_t for the client n, denoted by s_t^n, consists of $2(C+1)$ elements and is given as:

$$\begin{aligned} s_t^n = \quad & \{f_{t,s}^n(r_n), f_{t,s}^n(\mathcal{C}_n(1)), \ldots, f_{t,s}^n(\mathcal{C}_n(C)), \cdots \\ & f_{t,l}^n(r_n), f_{t,l}^n(\mathcal{C}_n(1)), \ldots, f_{t,l}^n(\mathcal{C}_n(C))\}, \end{aligned} \tag{15}$$

where $f_{t,x\in\{s,l\}}^n(v)$ denotes the view count of the video v for the client n during the last $L_{x\in\{s,l\}}$ video view times and $f_{t,s}^n(v) \le L_s$, $f_{t,l}^n(v) \le L_l$. $f_{t,s}^n(v)$ and $f_{t,l}^n(v)$ represent the frequency of the video v for a short-term period and a long-term period, respectively; thus, $L_s < L_l$. Each client updates its cache by replacing one video stored in its cache with the requested video r_n or keeps the cache as it is. Thus, a_t denoting an action that each client can take is defined as $a_t \in \mathcal{A} = \{0, 1, 2, \ldots, C\}$. The video $\mathcal{C}(a_t)$ will be replaced by r_n if $1 \le a_t \le C$. $a_t = 0$ denotes that the cache will be kept in its current state, which leads to $|\mathcal{A}| = C + 1$. The output $\pi(s_t)$ has the same size as $\mathcal{A}$. Contrary to conventional AC networks that choose a single action, the proposed *K*-AC selects the K elements with the largest value in $\pi(s_t)$ as candidate actions, which are denoted by $\hat{\boldsymbol{a}}_{\boldsymbol{t}} = \{\hat{a}_t^k | \hat{a}_t^k \in \mathcal{A}, 1 \le k \le K\}$. If $K = 1$, the *K*-AC becomes a conventional AC

network. $\hat{\boldsymbol{a}}_{\boldsymbol{t}}$ generates the set of K next states $\hat{\boldsymbol{s}}_{t+1} = \{\hat{s}^k_{t+1} | 1 \leq k \leq K\}$. The main value network evaluates the values of $\hat{\boldsymbol{s}}_t$ and $\hat{\boldsymbol{s}}_{t+1}$ by yielding $\boldsymbol{V}(\hat{\boldsymbol{s}}_{\boldsymbol{t}})$ and $\boldsymbol{V}(\hat{\boldsymbol{s}}_{\boldsymbol{t+1}})$, respectively, and the final action is selected as $a_t = \hat{a}^{k^*}_t$, where $k^* = \underset{k \in \{1,\cdots,K\}}{\arg\max}\, \boldsymbol{V}(\hat{\boldsymbol{s}}_{\boldsymbol{t+1}})$, while the corresponding next state is determined by $s_{t+1} = \hat{s}^{k^*}_{t+1}$. We designed rewards for our neural network in each client to minimize the number of transmissions per each client's video view. The rewards for each client are defined as:

$$r_t = \begin{cases} 1 & \text{for LC} \\ 0.5 & \text{for MC or XC}, \\ 0 & \text{for UC} \end{cases} \tag{16}$$

where LC has the largest reward because it requires no video transmissions, MC and XC have the second largest and the same reward because they can reduce the number of video transmissions by sharing network resources with other clients, and UC has the lowest reward because it cannot reduce the number of video transmissions. The number of transmissions might be a better reward than that in (16) because our goal was to reduce the number of transmissions. However, the proposed learning model was designed to be trained and run in a fully distributed manner without information exchange with other devices or the server, and it is thus impossible for each client to know the final number of transmissions. We used a replay memory and the concept of mini batch to train our networks by updating the parameters of the actor and main value networks, as depicted in Figure 4. The size of the mini batch is B. Through a back propagation, the parameters of the main value network are updated first, and those of the actor network are then updated. The parameters of the main value network are trained by using the B random samples to minimize the loss, which is defined as:

$$L_V := \frac{1}{B} \sum_{i=1}^{B} (r^i_t + \gamma \cdot V'(s^i_{t+1}) - V(s^i_t))^2, \tag{17}$$

where γ, denoting a discount factor, satisfies $0 \leq \gamma \leq 1$ and $V'(s^i_{t+1})$ is the output of the target value network. The target value network is used to generate the target Q-values for computing the loss during training and to keep the network from being destabilized by falling into feedback loops between the target and estimated Q-values. The parameters of the target value network are fixed and periodically updated by being replaced by those of the main value network. The parameters of the main value network, θ^V, are updated by the following gradient descent method:

$$\theta^V \leftarrow \theta^V + \alpha \nabla_{\theta^V} L_V, \tag{18}$$

where α denotes a learning rate. The loss function of the actor network is defined as:

$$L_A := \frac{1}{B} \sum_{i=1}^{B} \log \pi(a^i_t \mid s^i_t) A(s_t, a_t), \tag{19}$$

where $A(s_t, a_t)$ denotes the advantage function of the actor network and can be calculated as:

$$A(s_t, a_t) = r_t + \gamma \cdot V'(s_{t+1}) - V(s_t). \tag{20}$$

Finally, the parameters of the actor network θ^π are also updated by the gradient ascent method as follows:

$$\theta^\pi \leftarrow \theta^\pi + \beta \nabla_{\theta^\pi} L_A, \tag{21}$$

where β denotes a learning rate.

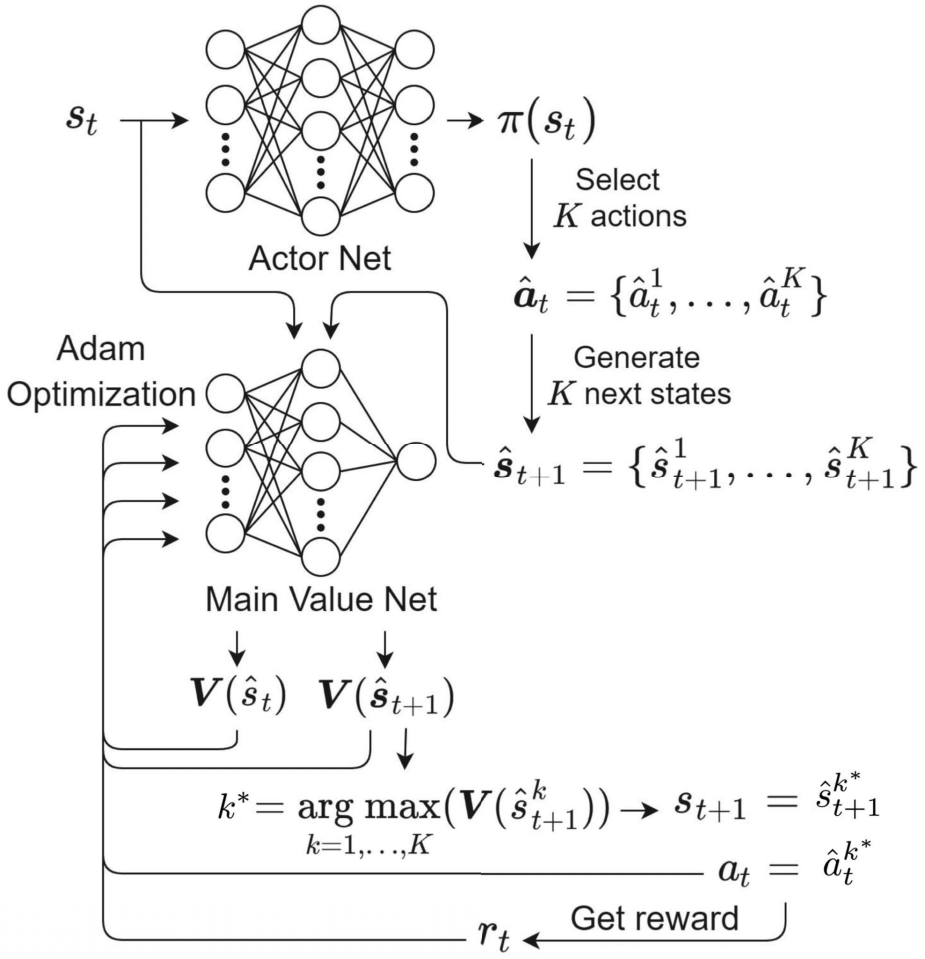

Figure 3. The proposed architecture of the *K*-AC.

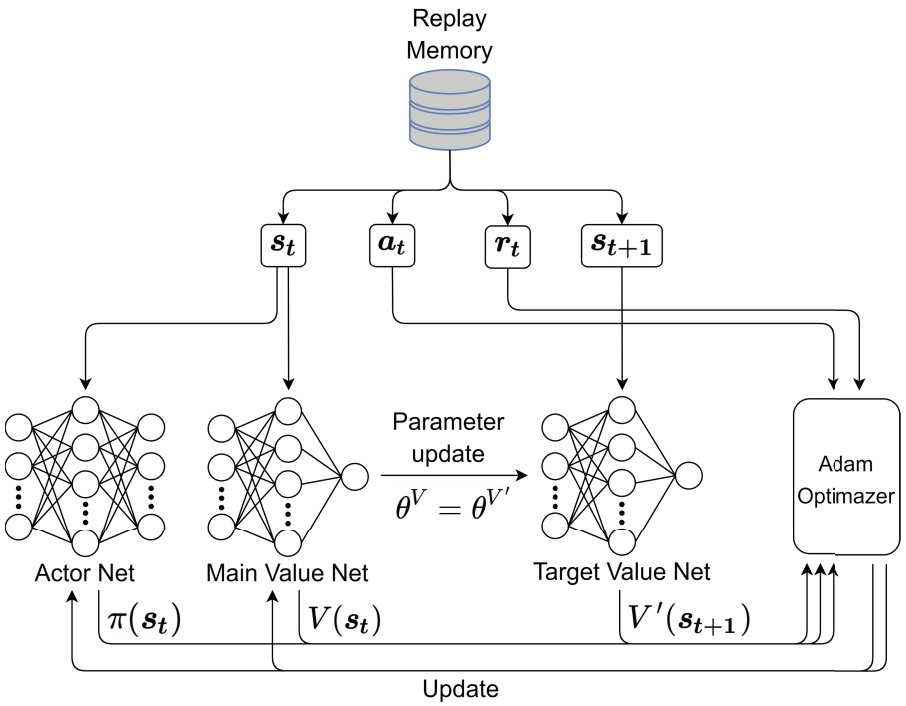

Figure 4. Illustration of the training process of the *K*-AC.

5. Numerical Results

In this section, we analyze the efficiency of the proposed cache update scheme using the K-AC in terms of the average number of transmissions per video streaming per client, which is defined as:

$$\eta = \mathbb{E}\left[\frac{K_{\mathrm{MC}} + K_{\mathrm{XC}} + K_{\mathrm{UC}}}{N}\right], \tag{22}$$

and compare it to that of conventional cache update schemes for both XC and non-XC. $0 \leq \eta \leq 1$, where $\eta = 0$ if all videos are transmitted through LC, while $\eta = 1$ if videos are all transmitted through UC. In the K-AC, the actor network consists of input, hidden, and output layers of sizes $2(C+1)$, $4(C+1)$, and $(C+1)$, respectively. The hidden layer is fully connected with the input and output layers. The ReLU and softmax functions are used as the activation functions for the input and hidden layers, respectively [24]. The value networks are the same as the actor network except that the output size is one. All parameters for the actor and value networks were initialized by He Uniform [25] and then updated iteratively by the Adam optimizer [26]. In our simulations, B and γ were set to 10 and 0.9, respectively, and L_s and L_l were set to 10 and 100, respectively. We compared the performance of the proposed K-AC with that of conventional cache update algorithms such as LRU, LFU, and FIFO, where it was assumed that $K = 10$.

Figure 5 shows the reward that the proposed K-AC scheme earns during a training process. p, denoting the probability that the popularity of videos changes, was set to 0.001, and the correlation factor ρ was set to 0.5. V, N, and β, denoting the number of videos, the number of clients, and the parameter of the Zipf distribution, were set to 100, 50, and 1, respectively. C, denoting the size of the cache, was set to 10 or 20. It is shown that the reward for $C = 10$ stabilized faster than for $C = 20$. More specifically, the reward for $C = 10$ stabilized after about 20 iterations, whereas the reward for $C = 20$ stabilized after about 40 iterations.

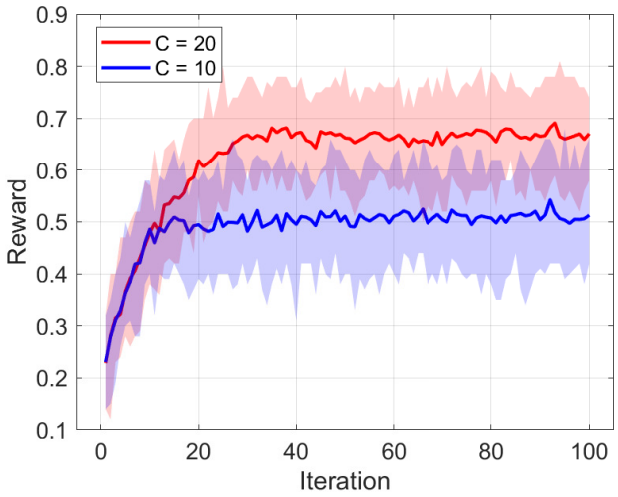

Figure 5. The rewards of the proposed scheme earned during a training process. $p = 0.001$, $\rho = 0.5$, $V = 100$, $N = 50$, $K = 10$, and $\beta = 1$.

Figures 6–10 show the average number of required transmissions per view of the video per client, defined in (22), for ρ, C, N, and β, respectively. According to Figure 6, the XC video stream scheme outperformed the non-XC scheme regardless of the cache update algorithms. The non-XC scheme denotes the conventional video streaming with UC and MC without supporting XC. As ρ decreased, videos' popularity changed less, and the average number of transmissions required for each video streaming decreased for all schemes. The proposed cache update scheme outperformed all conventional cache update schemes regardless of the value of ρ. For $\rho = 0.6$, the XC video stream scheme reduced η by about 23.2%, 23.7%, and 23%, compared to FIFO, LFU, and LRU, respectively. In addition,

the proposed cache update scheme could reduce η by about 8.8%, compared to LRU, which showed the best performance among the conventional schemes.

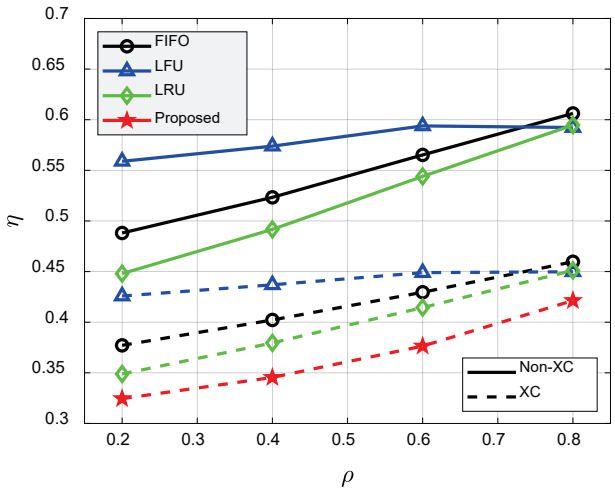

Figure 6. Average number of required transmissions for various ρ's. $p = 0.001$, $V = 100$, $N = 50$, $C = 20$, $K = 10$, and $\beta = 1$.

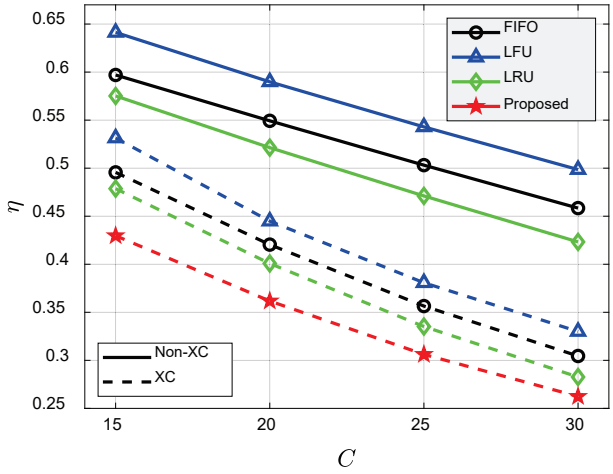

Figure 7. Average number of required transmissions for various C's. $p = 0.001$, $\rho = 0.5$, $V = 100$, $N = 50$, $K = 10$, and $\beta = 1$.

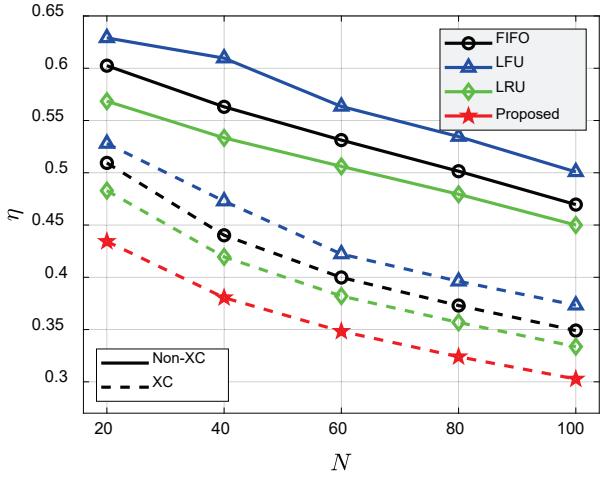

Figure 8. Average number of required transmissions for various N's. $p = 0.001$, $\rho = 0.5$, $V = 100$, $C = 20$, $K = 10$, and $\beta = 1$.

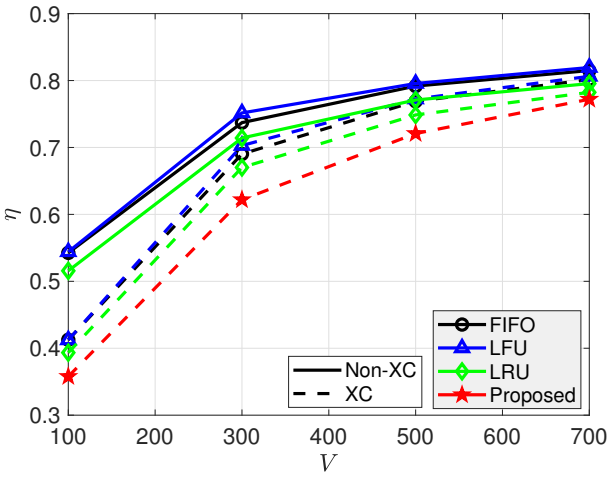

Figure 9. Average number of required transmissions for various V values. $p = 0.001$, $\rho = 0.5$, $\beta = 1$, $N = 50$, $C = 20$, and $K = 10$.

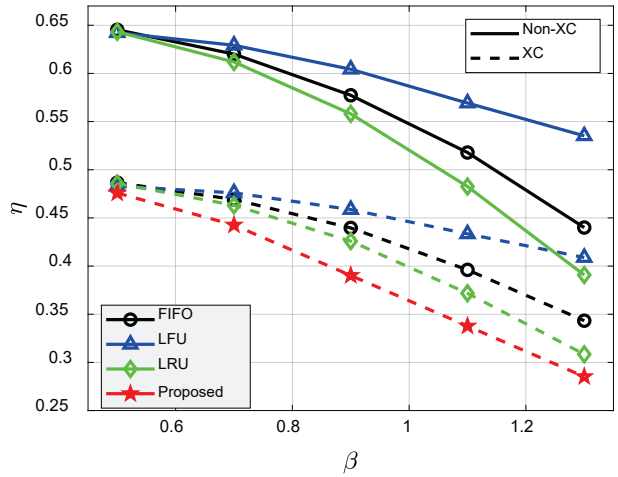

Figure 10. Average number of required transmissions for various β's. $p = 0.001$, $\rho = 0.5$, $V = 100$, $N = 50$, $C = 20$, and $K = 10$.

Figures 7 and 8 show that η decreased as C or N increased for all schemes. The greater the C, the more the LC was because the probability that requested videos were already cached in the clients' cache increased. The greater the N, the more the MC or XC was where multiple videos can be transmitted by single transmission. In addition, the XC video streaming scheme outperformed the non-XC video streaming scheme for all cache update schemes, and the proposed cache update scheme based on K-AC yielded the best performance. In Figure 7, when $C = 15$, the XC video streaming scheme could reduce η by about 16.5%, 16.7%, and 16.3%, compared to FIFO, LFU, and LRU, respectively, and the proposed cache update scheme could reduce η by about 9.9% compared to LRU, which yielded the best performance among the conventional schemes. In Figure 8, when $N = 20$, the XC video streaming scheme could reduce η by about 18.6%, 15.6%, and 14.6%, compared to FIFO, LFU, and LRU, respectively, and the proposed cache update scheme could reduce η by about 9.7%, compared to LRU.

Figure 9 shows η for various V values. For constant C and N, the possibility of MC and XC decreased as V increased, and η thus decreased for all schemes as V increased. The proposed cache update scheme outperformed all conventional schemes for all V values. Finally, Figure 10 shows that η decreased as β increased because clients were inclined to request highly popular videos, and the probability of LC, MC, or XC also increased. For $\beta = 0.9$, the XC video streaming scheme reduced η by about 23.1%, 23.4%, and 22.9%, compared to FIFO, LFU, and LRU, respectively, and the proposed cache update scheme

could reduce η by about 8%, compared to LRU, which showed the best performance among the conventional schemes.

6. Conclusions

In this work, we investigated a cache management problem for XC video streaming systems, where each client needs to update its cache so as to increase the probability of XC with other clients, as well as its own hit probability, while each client's hit probability has been only considered in conventional video streaming systems. We formulated a cache management problem for XC video streaming systems and investigated how to minimize the number of XOR operations. We also proposed how to update the clients' cache to improve the efficiency of video streaming by decreasing the number of transmissions. Contrary to most existing studies assuming that all clients have the same popularity of videos and the popularity is time invariant, our study considered that the popularity varies over time and is differently distributed for each client. Based on these practical assumptions, we proposed a new cache update scheme using reinforcement learning. The proposed scheme used the K-AC network to overcome the disadvantages of conventional AC networks. Each client can train its own K-AC network by using the local information, which does not require any feedback or signaling, and can decide whether to update its cache. If a client decides to update its cache, the video to be replaced by a new one is decided by the action of the K-AC. Thus, the proposed scheme is completely decentralized. We analyzed the performance of the proposed scheme in terms of the average number of required transmissions per each video streaming per client, which was compared to that of conventional cache update schemes such as FIFO, LFU, and LRU. Our numerical results showed that XC video streaming outperformed non-XC video streaming, and the proposed cache update scheme using the K-AC yielded the best performance. Specifically, when $V = 100$, $N = 50$, $C = 15$, and $\beta = 1$, the ρ's for non-XC LRU, XC LRU, and the proposed scheme were 0.58, 0.48, and 0.44, respectively. Thus, it can be concluded that the proposed scheme could reduce the number of transmissions by 24.1% and 8.3%, compared to the non-XC LRU and XC-LRU schemes, respectively.

Author Contributions: Conceptualization and problem formulation, T.-W.B.; writing—original draft preparation, Y.-S.K.; methodology and formal analysis, J.-M.L.; visualization and simulations, Y.-S.K.; funding acquisition, T.-W.B. writing—review and editing, J.-Y.R. All authors 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).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Cisco. Cisco visual networking index: Global mobile data traffic forecast update, 2017–2022. *Update* **2019**, *2017*, 2022.
2. Gill, P.; Arlitt, M.; Li, Z.; Mahanti, A. YouTube Traffic Characterization: A View From the Edge. In Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement 20067, San Diego, CA, USA, 24–26 October 2007.
3. Jeon, J. NR Wide Bandwidth Operations. *IEEE Commun. Mag.* **2018**, *56*, 42–46. [CrossRef]
4. Ngo, H.Q.; Tran, L.; Duong, T.Q.; Matthaiou, M.; Larsson, E.G. On the Total Energy Efficiency of Cell-Free Massive MIMO. *IEEE Trans. Green Commun. Netw.* **2018**, *2*, 25–39. [CrossRef]
5. Ge, X.; Yang, J.; Gharavi, H.; Sun, Y. Energy Efficiency Challenges of 5G Small Cell Networks. *IEEE Commun. Mag.* **2017**, *55*, 184–191. [CrossRef] [PubMed]
6. Poularakis, K.; Iosifidis, G.; Sourlas, V.; Tassiulas, L. Exploiting Caching and Multicast for 5G Wireless Networks. *IEEE Trans. Wirel. Commun.* **2016**, *15*, 2995–3007. [CrossRef]

7. Kanrar, S.; Mandal, N.K. Traffic analysis and control at proxy server. In Proceedings of the 2017 International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 15–16 June 2017; pp. 164–167.
8. Bidokhti, S.; Wigger, M.; Timo, R. Noisy broadcast networks with receiver caching. *IEEE Trans. Inf. Theory* **2018**, *64*, 6996–7016. [CrossRef]
9. 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]
10. 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, Turkey, Sunday, 7–12 July 2013; pp. 962–966.
11. 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]
12. 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]
13. Ban, T.-W.; Lee, W.; Ryu, J. An efficient coded streaming using clients' cache. *Sensors* **2020**, *20*, 6220. [CrossRef] [PubMed]
14. Ascigil, O.; Phan, T.K.; Tasiopoulos, A.G.; Sourlas, V.; Psaras, I.; Pavlou, G. On uncoordinated service placement in edge-clouds. In Proceedings of the 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Hong Kong, China, 11–14 December 2017; pp. 41–48.
15. Aimtongkham, P.; So-In, C.; Sanguanpong, S. A novel web caching scheme using hybrid least frequently used and support vector machine. In Proceedings of the 2016 13th International Joint Conference on Computer Science and Software Engineering (JCSSE), Khon Kaen, Thailand, 13–15 July 2016; pp. 1–6.
16. Nogueira, J.; Gonzalez, D.; Guardalben, L.; Sargento, S. Over-The-Top Catch-up TV content-aware caching. In Proceedings of the 2016 IEEE Symposium on Computers and Communication (ISCC), Messina, Italy, 27–30 June 2016; pp. 1012–1017.
17. O'neil, E.; O'Neil, P.; Weikum, G.; Zurich, E. The LRU–K Page Replacement Algorithm For Database Disk Buffering. *ACM SIGMOD Rec.* **1993**, *22*, 297–306. [CrossRef]
18. Abrahamsson, H.; Björkman, M. Caching for IPTV distribution with time-shift. In Proceedings of the 2013 International Conference on Computing, Networking and Communications (ICNC), San Diego, CA, USA, 28–31 January 2013; pp. 916–921.
19. Abdelkrim, E.; Salahuddin, M.A.; Elbiaze, H.; Glitho, R. A Hybrid Regression Model for Video Popularity-Based Cache Replacement in Content Delivery Networks. In Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM), Washington, DC, USA, 4–8 December 2016; pp. 1–7.
20. Maniotis, P.; Thomos, N. Viewport-Aware Deep Reinforcement Learning Approach for 360° Video Caching. *IEEE Trans. Multimed.* **2021**, to be published. [CrossRef]
21. Zhong, C.; Gursoy, M.C.; Velipasalar, S. A deep reinforcement learning-based framework for content caching. In Proceedings of the 2018 52nd Annual Conference on Information Sciences and Systems (CISS), Princeton, NJ, USA, 21–23 March 2018; pp. 1–6. [CrossRef]
22. Dulac-Arnold, G.; Evans, R.; van Hasselt, H.; Sunehag, P.; Lillicrap, T.; Hunt, J.; Mann, T.; Weber, T.; Degris, T.; Coppin, B. Deep reinforcement learning in large discrete action spaces. *arXiv* **2015**, arXiv:1512.07679.
23. Breslau, L.; Cao, F.; Phillips, G.; Shenker, S. Web caching and Zipf-like distributions: Evidence and implications. In Proceedings of the IEEE INFOCOM '99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No.99CH36320), New York, NY, USA, 21–25 March 1999; Volume 1, pp. 126–134.
24. Wang, Y.; Li, Y.; Song, Y.; Rong, X. The Influence of the Activation Function in a Convolution Neural Network Model of Facial Expression Recognition. *Appl. Sci.* **2020**, *10*, 1897. [CrossRef]
25. Meißner, P.; Watschke, H.; Winter, J.; Vietor, T. Artificial Neural Networks-Based Material Parameter Identification for Numerical Simulations of Additively Manufactured Parts by Material Extrusion. *Polymers* **2020**, *12*, 2949. [CrossRef] [PubMed]
26. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations, New Orleans, LA, USA, 12 December 2014.