future internet

MDPI

Article

Distributed Bandwidth Allocation Strategy for QoE Fairness of Multiple Video Streams in Bottleneck Links

Yazhi Liu, Dongyu Wei, Chunyang Zhang and Wei Li *

College of Artificial Intelligence, North China University of Science and Technology, Tangshan 063210, China; liuyazhi@ncst.edu.cn (Y.L.); 15638005630@163.com (D.W.); zhangchunyang0612@163.com (C.Z.)
* Correspondence: lw@ncst.edu.cn

Abstract: In QoE fairness optimization of multiple video streams, a distributed video stream fairness scheduling strategy based on federated deep reinforcement learning is designed to address the problem of low bandwidth utilization due to unfair bandwidth allocation and the problematic convergence of distributed algorithms in cooperative control of multiple video streams. The proposed strategy predicts a reasonable bandwidth allocation weight for the current video stream according to its player state and the global characteristics provided by the server. Then the congestion control protocol allocates the proportion of available bandwidth, matching its bandwidth allocation weight to each video stream in the bottleneck link. The strategy trains a local predictive model on each client and periodically performs federated aggregation to generate the optimal global scheme. In addition, the proposed strategy constructs global parameters containing information about the overall state of the video system to improve the performance of the distributed scheduling algorithm. The experimental results show that the introduction of global parameters can improve the algorithm's QoE fairness and overall QoE efficiency by 10% and 8%, respectively. The QoE fairness and overall QoE efficiency are improved by 8% and 7%, respectively, compared with the latest scheme.

Keywords: QoE fairness; video quality; federated learning; deep reinforcement learning; congestion control

Citation: Liu, Y.; Wei, D.; Zhang, C.; Li, W. Distributed Bandwidth Allocation Strategy for QoE Fairness of Multiple Video Streams in Bottleneck Links. *Future Internet* **2022**, *14*, 152. https://doi.org/10.3390/fi14050152

Academic Editors: Choong Seon Hong and Latif U. Khan

Received: 10 April 2022
Accepted: 16 May 2022
Published: 18 May 2022

1. Introduction

In recent years, video traffic in the network has shown a rapid growth trend, mainly from HTTP streaming media and video-on-demand services, which aim to provide users with the highest quality of experience (QoE). Therefore, many previous studies have focused on designing a more powerful adaptive bitrate algorithm (ABR) to improve the QoE of a single video user.

With the growth of video traffic and network connection equipment, the possibility of multiple video streams competing for bandwidth in the same bottleneck link in the network is becoming higher and higher [1]. Therefore, QoE fairness among multiple video users has become a crucial issue.

At present, the fairness scheduling scheme for commercial video services across multiple users still focuses on allocating the same bandwidth for each user as much as possible. That is, each video stream independently makes adaptive bitrate decisions through its own ABR algorithm and then divides the bandwidth for each video stream in the bottleneck link as equally as possible by the congestion control protocol. Unfortunately, although these schemes can allocate the same bandwidth to each user, QoE fairness among video users cannot be ensured due to the viewing conditions of different users and the differences in on-demand video content.

To study the impact of differences in video content on perceptual quality, we randomly sampled videos from two different genres (science fiction and animation) and measured their average perceptual video quality at several different bitrates. The results are shown in Figure 1. Under the same bandwidth conditions, users of video 2 obtain higher viewing quality than users of video 1.

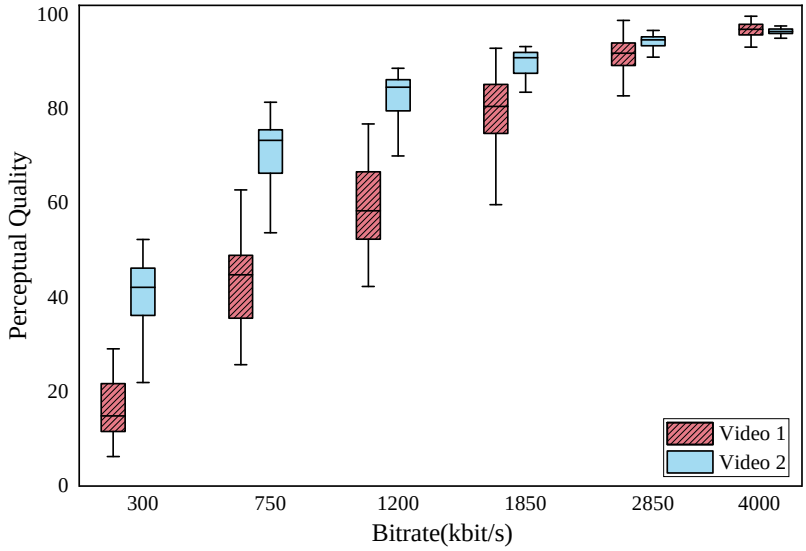


Figure 1. The perceived quality of different videos at each bitrate.

The researches attempted to improve QoE fairness among multiple video streams by actively allocating bandwidth to each video stream in recent years. Although this method has achieved good results, it still faces the following problems:

(1) Because of the complexity and time variability of the network, it is difficult to determine when congestion occurs among streams. Bandwidth allocation may lead to a decline in link utilization;
(2) Because of the instability of multiple video streams, the behavior of each video stream affects the playback state of other video streams, which may lead to the algorithm being unable to learn the optimal strategy or even losing the guarantee of convergence.

A recent trend is to solve related problems in the field of video streaming through deep reinforcement learning (DRL). The video system itself can provide a large number of optional parameters, and the deep reinforcement learning algorithm can use a variety of parameters as input signals through a large amount of training to find the law between the data so that the algorithm has good performance. Therefore, it has specific applicability to deal with the instability of the multiple video stream environment in the field of video streaming.

However, the DRL algorithm strongly depends on the training environment and training data. Machine learning algorithms, which train a large number of representative data, are often a more effective method. It is difficult for the existing machine learning methods to obtain enough representative training data in the related fields of video streaming. This is because the network tracking data used to simulate the Internet environment are collected from a single node in the network. However, the Internet itself is highly complex and diverse. What any single node observes is only a tiny part of the complex behavior of the network. Therefore, it is impossible to simulate the complex and diverse Internet [2]. For this reason, although the algorithm trained in the simulation environment has a good performance in the simulator, it is often unsatisfactory when deployed to the real network.

In response to this problem, Yan et al. [3] trained the algorithm by using a large amount of telemetry data of real video clients collected from video websites, so the algorithm also has good performance when extended to the real network. However, this scheme requires that all client data be periodically centralized on the server for centralized training without considering the additional communication burden caused by the transmission of a large amount of training data, which may have a negative impact on the quality of video transmission.

In this paper, we aim to design a fair bandwidth allocation based on congestion control, which can maximize the QoE fairness among video users while maintaining the overall

QoE efficiency and network bandwidth utilization of the video system. In general, the contributions of this paper are as follows:

(1) An active bandwidth allocation framework based on congestion control is developed, which can ensure the utilization of network links while allocating bandwidth;
(2) Combined with the characteristics of the video on demand system, a parameter containing a global state is constructed as the input of the neural network, which effectively improves the performance of the distributed DRL algorithm in multiple video stream scenes;
(3) The combination of deep reinforcement learning and federated learning [4] is used to optimize the QoE fairness of multiple video streams. A video system is constructed in the real network to complete the deployment and evaluation of the algorithm.

2. Related Work

This section summarizes the QoE fairness of adaptive video streams and multiple video streams, and introduces the latest research progress in video stream fairness.

In the traditional dynamic adaptive streaming over HTTP, the video is divided into multiple blocks (generally 2–8 s in length). Each video block is independently encoded at different bitrate levels. Therefore, the client can switch the video to another bitrate at the boundary of any video block. ABR algorithm can determine the bitrate of the video block to be acquired without determining the available bandwidth in the future to maximize the quality of the video block, reduce the switching frequency of the bitrate, and reduce the occurrence of the playing caton phenomenon as much as possible. The importance of these factors is measured by the QoE index of a single user.

In optimizing single video streams, each stream is only optimized within its own available bandwidth, regardless of how the bandwidth share among video streams is allocated. However, the QoE fairness of multiple video streams should consider the following two optimization spaces: (1) The impact of different video contents on the quality of user experience. Different videos need different bandwidths to achieve the same viewing quality. Ideally, players with low video quality should get more bandwidth than players with high video quality. (2) Different states of different video players. For example, players with low buffer levels should obtain more bandwidth than players with high buffer levels to reduce the risk of system re-buffering while maximizing the overall QoE efficiency.

Although better ABR algorithms can improve the fairness of video streaming to some extent, such as RESA [5] and Bola [6], these schemes only attempt to provide the same bandwidth for each video user, rather than the equitable allocation of QoE among users. To solve this problem, it is necessary to allocate a more reasonable bandwidth for each video stream in the bottleneck link. Existing solutions can be roughly divided into the following two categories.

Centralized Bandwidth Allocation: This type of scheme typically has a video server or bottleneck router that computes a bandwidth allocation that optimizes QoE fairness for each video stream based on the state of each video stream and enforces that bandwidth allocation during video transmission. For example, Yin et al. [7] allocate bandwidth to each video stream to improve QoE fairness by deploying a controller in bottleneck routing. However, this scheme requires deploying a network controller on each router in the network, with the controller being able to access the QoE information of all video streams traversing this link; this is difficult for large-scale deployments. In addition, Altamimi et al. [8] used the network packet loss rate to evaluate the status of each video stream. Then, the video server limited the highest bitrate that the video stream could request according to status and used this to achieve all the allocation of available bandwidth for video streaming. However, this bandwidth allocation scheme may decrease link utilization because the server cannot accurately locate the congestion.

Decentralized bandwidth allocation: Nathan et al. [9] constructed a utility function using the historical QoE of video users and the estimated value of future QoEs, and dynamically configured the parameters of the congestion protocol for each video stream through the utility

function. The distributed method allocates bandwidth for each video stream in the bottleneck link, improving QoE fairness and ensuring complete link utilization. However, this scheme needs to calculate the QoE of each video block in all possible states in advance during downloading, thus incurring huge computational overhead. In response to the above problems, Jiang et al. [10] used the historical QoE of video streams to estimate the future QoE, which reduced the computational overhead of utility functions and the difficulty of algorithm deployment. However, this scheme relies more on the accuracy of bandwidth estimation, and the performance is degraded in a highly dynamic network environment.

In another important work in the field of multi-stream optimization is the definition of video perceptual quality, Jiang et al. [10] used a linear combination of screen size and resolution as the definition of perceptual video quality. In addition, VMAF and PSNR use video content-specific features to calculate perceptual quality scores and thus are often used in the field of video streaming optimization. The QoE definition in this paper can support any video quality metric.

3. System Overview

Based on the above points, we design a fair bandwidth allocation based on the congestion control (FBAC) algorithm for QoE fairness.

3.1. Problem Description

In this paper, we consider that n concurrent video sessions in the network share the same bottleneck link. Each video session consists of a global sender (video server) and a separate receiver (video player). The receiver has an independent ABR algorithm for determining its bitrate.

The player $i \in N(N = \{1, \ldots, n\})$ starts at a random point in time t_i and requests a random video in the video dataset, where each video is independently encoded at the same set of bitrate levels R. When downloading video, the player i selects the bitrate level $r_i(k_i) \in R$ for the k_i-th video block through its own ABR algorithm.

3.1.1. Definition of QoE

In this paper, QoE is modeled as a linear combination of three factors: perceptual quality, smoothness, and rebuffering time of a video block:

$$Q_i = q_i(r_i[k_i]) - \alpha |q_i(r_i[k_i]) - q_i(r_i[k_i - 1])| - \beta re_i(k_i) \tag{1}$$

where $q_i(r_i[k_i])$ is used to represent the perceived quality that can be obtained by viewing the k_i-th block. It supports any index that can predict the perceived quality of users according to the video content. In order to distinguish the difference in viewing quality caused by different video contents, FBAC uses VMAF [11] to estimate the perceived quality of video blocks; $q_i(r_i[k_i]) - q_i(r_i[k_i - 1])$ represents the change of perceived quality between adjacent blocks, that is, the smoothness of video; α represents the penalty coefficient of smoothness. The larger α is, the less the users can accept the changes of video quality; $re_i(k_i)$ represents the re-buffering time experienced while viewing the k_i-th block; β represents the penalty coefficient of the re-buffered event.

3.1.2. QoE Fairness

In previous work, Jain's fairness coefficient was widely used to describe the fairness state of QoE. Therefore, we use Jain's fairness coefficient and historical discount QoE to calculate the QoE fairness of multiple video streams over a period of time:

$$J(QoE_1^{\gamma,h}, QoE_2^{\gamma,h}, \ldots, QoE_n^{\gamma,h}) = \frac{(\sum_{i=1}^{n} QoE_i^{\gamma,h})^2}{n \cdot \sum_{i=1}^{n} (QoE_i^{\gamma,h})^2} \tag{2}$$

$$QoE_i^{\gamma,h} = \sum_{l=k-h}^{k} \gamma^l Q_i \tag{3}$$

where $QoE_i^{\gamma,h}$ represents the QoE historical discount score of the *i-th* video stream; γ is the discount coefficient of QoE; h represents the number of historical blocks to be calculated. When $h = 1$, it means that only the short-term QoE fairness of the current video block is calculated. When $h = \infty$, it means that the overall QoE fairness of all blocks is calculated.

3.2. Bandwidth Allocation Framework Based on Congestion Control

When multiple video streams are in the same bottleneck link, the available capacity of the bottleneck link cannot be changed by any other methods. The QoE fairness between videos can only be improved by adjusting the proportion of video streams in the throughput of the bottleneck link. In this paper, we attempt to predict the reasonable weight of bandwidth allocation W_k^i for each video stream through the deep reinforcement learning algorithm and then use the congestion control protocol to allocate the bandwidth matching the weight for each video stream in the bottleneck link.

Figure 2 depicts the basic flow of a distributed active bandwidth allocation framework based on congestion control. Firstly, the ABR algorithm of the client i makes bitrate decisions based on the local observation value S_i^2 of the player. Meanwhile, the QoE of each video block is calculated according to Equation (1) and sent to the video server. The server can generate the parameter S_i^1 containing global characteristics according to the QoE information of all clients currently connected to it. Then, the DRL algorithm on the client predicts the weight of bandwidth allocation W_k^i of the current video stream according to S_i^1 and S_i^2. The server needs to continuously monitor the QoE information of each client to generate global parameters, and reconfigure the relevant parameters of congestion control for the video stream i to allocate bandwidth according to the weight W_k^i predicted by the client.

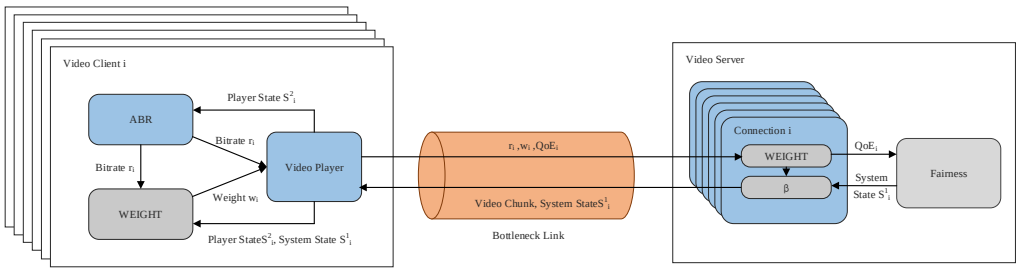


Figure 2. Active bandwidth allocation framework based on congestion control.

It is worth noting that FBAC attaches the parameters (QoE, weight, etc.) that need to be passed between the client and the server to the header of the HTTP request in order to reduce the communication overhead of the system:

(1) The client s sends a GET request for the *k-th* video block to the server, and the request header is accompanied by the bandwidth allocation weight W_k^i of the *k-th* block and the QoE score $QoE_{(k-1)^i}$ of the $k-1$-*th* block;
(2) The server receives and parses the information W_k^i, $QoE_{(k-1)^i}$ attached to the header in the request;
(3) The server updates the congestion control parameters of the video stream and generates global state parameters $S_{k^i}^1$ according to $QoE_{(k-1)^i}$ at the same time;
(4) The server returns the *k-th* video block to the client with the global status parameter $S_{k^i}^1$ in the response header;
(5) The client plays the video block *k-th* and calculates the QoE_{k^i} that can be obtained by playing the video block, and the weight prediction module predicts the band-

width allocation weight W_{k+1}^{i} during the download of the $k+1$-th block according to the parameters, including the parameter S_{ki}^{1};

(6) The client issues a GET request for the $k+1$-th block.

For the fairness algorithm to work, FBAC must also use the existing congestion control algorithm to convert the weight value wi of each video stream into the available bandwidth proportional. Here, we discuss how to use packet loss-based CUBIC [12] and delay-based FAST [13], two types of congestion control, to allocate user bandwidth shares. In this paper, the cube implemented in the QUIC protocol is used as the underlying congestion control for algorithm deployability.

Cubic will reduce the size of the sending window after detecting packet loss, and the magnitude of the reduction is related to the multiplicative subtraction factor β of Cubic. Therefore, by adjusting the multiplicative subtraction factor β_i corresponding to each stream, the video in the bottleneck link can be obtained. This is the percentage of available bandwidth for the stream allocation. When CUBIC is in a deterministic loss model with packet loss rate p, the relationship between Cubic's average sending window size AVG_C and the multiplicative subtraction factor β can be expressed by the following function [14]:

$$AVG\_C = 0.4\left(\frac{3+\beta}{4(1-\beta)}\right)^{\frac{1}{4}}\left(\frac{RTT}{p}\right)^{\frac{3}{4}} \tag{4}$$

where p is the packet loss rate, and RTT is the round-trip time.

For delay-based congestion control FAST, the video stream can set the congestion window size to:

$$cwnd^{'} = \left(cwnd\frac{RTT}{RTT_{min}} + w_i\right) \tag{5}$$

Research has proved that this method can also achieve bandwidth allocation proportional to the weight w_i .

3.3. Multi-Agent Reinforcement Learning

In this study, each video client running in the network is regarded as an independent agent. These agents regard other agents as a part of the environment. Each agent selects actions and obtains rewards only based on its own observations. Although this method is easy to deploy and decentralized, it also brings some problems. For a given agent $i \in \{1,\ldots,n\}$, there is no way to distinguish the randomness of the environment from the behavior of other agents $i \in \{1,\ldots,i-1,i+1,\ldots,n\}$. In addition, because the reward R_i is not all from the actions performed by the agent, it is difficult to assign the correct reward to each action agent, resulting in an inability to learn the optimal strategy.

Ryan et al. [15] found that when Independent Proximal Policy Optimization (IPPO) uses more comprehensive global features as the input of the neural network, it can often obtain better performance than QMix [16]. Whether the input of the value network contains the characteristics of the global state is the fundamental difference between Multi-Agent Reinforcement Learning and completely independent Distributed Reinforcement Learning. It is essential for the DRL algorithm in the multi-agent scenario to ensure that the input parameters of the neural network contain enough global information [17].

To solve the convergence problem of an algorithm based on distributed bandwidth allocation and improve the algorithm's performance, in this paper, we attempt to make the neural network input contain as many global features as possible and reduce the extra communication cost from parameter transmission. Therefore, we constructed two global parameters counted by the server, namely coefficient of variation of QoE C and QoE level of client L, which are defined as follows:

$$C = \frac{\sigma_{QoE^{\gamma,h}}}{\mu_{QoE^{\gamma,h}}} \tag{6}$$

$$L = \frac{QoE_i^{\gamma,h}}{\frac{1}{n}\sum_{i=0}^{n} QoE_i^{\gamma,h}} \tag{7}$$

where $\sigma_{QoE^{\gamma,h}}$ represents the standard deviation of the historical discount QoE for the past h video blocks of all video clients served by the current server; $\mu_{QoE^{\gamma,h}}$ represents the mean value of the historical discount QoE of the past h video blocks of all video users. QoE coefficient of variation is used to indicate the dispersion degree of QoE score distribution of video stream users. The greater the value, the greater the dispersion degree. The generation of the above two parameters only involves QoE user information transmission. As described in Section 3.2, the transmission opportunity only occurs when the client sends a new GET request to the server. Therefore, the agent can obtain the overall state information of the current system with minimal communication and calculation cost.

3.4. Deep Reinforcement Learning Framework

In this paper, we use the Proximal Policy Optimization Algorithm (PPO) [18] to train the DRL neural network, a policy-based actor-critic algorithm. Each DRL agent must train a policy network π_θ and a value network v_ϕ simultaneously. The specific functions of the algorithm are as follows:

Status: observed value $O_i(QoE_i^{\gamma,h}, g_i, f_i, b_i, T_i, C_i, L_i)$, $QoE_i^{\gamma,h}$ represents the QoE score of historical discount of h video blocks played by agent i in the past; g_i represents the set of file sizes of six optional bitrates for the next video block; f_i represents the set of video quality scores of the six selectable bitrates for the next video block; b_i represents the current buffer level of agent i; T_i represents the current bandwidth estimation of agent i; C_i represents the current QoE coefficient of variation of agent i; L_i represents the current QoE level of the client of agent i.

Action: The action taken by the agent in time step t_i is the weight of bandwidth allocation $w_i \in [0.4, 0.9]$ of the video user during the next video block download. The w_i is modeled as a set of continuous variables to explore the nearly optimal strategy in the appropriate action space. The value range of w_i is limited to avoid excessive adjustment caused by a too large or too small retreat factor.

Reward: The common goal of all agents is to maximize overall QoE efficiency and ensure QoE fairness among users. Therefore, the reward R_i of each agent is defined as the linear combination of its own historical discount QoE score and QoE fairness index (Equation (1)):

$$R_i = QoE_i^{\gamma,h} + \delta J(QoE_1^{\gamma,h}, QoE_2^{\gamma,h}, \ldots, QoE_n^{\gamma,h}) \tag{8}$$

The loss function of the cumulative discount reward of Policy Network π_θ is:

$$\nabla_\theta L^n(\theta) = E_{o_t^n, a_t^n}\left[\nabla_\theta log\left\{min\Big(\frac{\pi_\theta(a_t^n|o_t^n)}{\pi_{\theta'}(a_t^n|o_t^n)}A(o_t^n,a_t^n), clip\Big(\frac{\pi_\theta(a_t^n|o_t^n)}{\pi_{\theta'}(a_t^n|o_t^n)}, 1-\xi, 1+\xi\Big)A(o_t^n,a_t^n)\Big)\right\}\right] \tag{9}$$

where π_θ is the old policy parameter before the update, $A(o_t^n, a_t^n)$ is the advantage function, $E_{o_t^n, a_t^n}$ represents the mathematical expectation of the loss function for all observation-action cases, and $clip$ is the clipping function: if the first term is less than the second term, output $1-\xi$; if the first term is greater than the third term, output $1+\xi$. ξ is a hyperparameter, which is set to 0.2 in this paper.

The update rule for the policy network π_θ is given by:

$$\theta' \leftarrow \theta + \varepsilon \nabla_\theta log\left\{min\Big(\frac{\pi_\theta(a_t^n|o_t^n)}{\pi_{\theta'}(a_t^n|o_t^n)}A(o_t^n,a_t^n), clip\Big(\frac{\pi_\theta(a_t^n|o_t^n)}{\pi_{\theta'}(a_t^n|o_t^n)}, 1-\xi, 1+\xi\Big)A(o_t^n,a_t^n)\Big)\right\} \tag{10}$$

The update rule of the value network v_ϕ is:

$$\phi' \leftarrow \phi + \varepsilon' \nabla_\phi L^n(\phi) \tag{11}$$

$$L^n(\phi) = E_{o_t^n}\Big[\bigtriangledown_\phi log\Big((V_\phi(o_t^n) - R_t^n)^2\Big)\Big] \tag{12}$$

where ε and $\varepsilon^{'}$ are the learning rates of the policy network and value network, respectively. v_ϕ is the value policy parameter, R is the discount reward. In this paper, the neural networks trained by each agent are homogeneous, and the same hyper-parameters are adopted in the training process. The values of ε and $\varepsilon^{'}$ are set to 0.005.

3.5. Training the Model Using Federated Learning

As a distributed learning framework, federated learning has the following advantages: First, each agent is trained on the local data set without centrally uploading a large amount of data to the server, which not only dramatically reduces the cost of communication, but also reduces the risk of privacy leakage. Second, the federated learning algorithm can deal with the problem of non-independent and identically distributed data and different amounts of proxy data. Third, although federated learning cannot surpass the centralized training model, it can still perform similarly. Based on the above reasons, we adopted the federated learning framework to train the DRL model, and the specific process is shown in Algorithm 1.

Algorithm 1 weight prediction model training algorithm based on federated learning

Initialize global policy network parameters θ and value network parameters ϕ
Initialize all agents $i \in N$
for $step = 1, 2, \ldots T$ **do**
  for each agent $i \in N$ **do**
    Download the model parameters from the server so that $\theta_i = \theta, \phi_i = \phi$
    Train the model with local data and update the model parameters $\theta_i^{'}, \phi_i^{'}$
    Randomly select m available agents to get the set of agents M_{step}.
    for agent $i \in M_{step}$ **do**
      Upload local model parameters $\theta_i^{'}$, $\phi_i^{'}$ to the server
    end for
  end for
  Service-Terminal:
  Receive all model parameters
  Perform model parameter aggregation, update global model parameters θ_{step}, ϕ_{step}
end for

4. Implementation

FBAC is implemented through a DASH video client and a video server running on QUIC. Each video client runs a weight prediction algorithm independently, and the video server can use the weight to adjust the user's available bandwidth through the underlying congestion control algorithm.

4.1. Video Client

In this paper, the developer version of the Google Chrome browser was used as the video client. The Chrome browser can be configured to establish a QUIC connection with the video server. Dash.js has been modified to track the parameter input of the DRL algorithm during video playback and the parameter passing between client and server. The video streaming service starts by running dash.js in each Chrome browser, and the MPC [19] algorithm makes bitrate decisions.

4.2. Video Server

In order to facilitate the modification of the network congestion control protocol, we adopted the Nginx server of QUIC protocol as the platform of network transmission, and it was deployed on the Alibaba cloud server. The video server can analyze the weight of bandwidth allocation attached to the GET request and update the multiplicative decrease

factor of each stream by improving the relevant modules of Nginx and the underlying congestion control code of quiche.

4.3. Neural Networks

To train the DRL agent, in this study, we stored the historical QoE score, the file size of the next video block, and the video quality score each in an array with a size of 6. These arrays pass through a 1D convolution layer composed of 32 filters. The results of the convolution layer are aggregated with the other four observations through a connecting layer and finally connected with the final softmax activation function through a hidden layer composed of 64 neurons. In addition, the value network uses the same neural network structure, but its final output is a linear neuron.

4.4. Datasets

Given the differences in the content of different videos, we selected videos from different genres and integrated them into a set of video datasets. Genres covered science fiction, animation, action, and other types. All videos are transcoded and sliced by H.264/MPEG-4 codec at {300 k, 750 k, 1200 k, 1850 k, 2850 k, 4000 k} bps bitrates, and the length of each video block is 5 s. In addition, the VMAF scores of all video blocks are calculated and stored on the server. The modified dash.js obtains the VMAF scores of the video block synchronously when the video block is requested.

5. Evaluation

FBAC has two design goals: (1) to maximize QoE fairness among users and (2) to improve the overall QoE efficiency of users. To accurately verify the performance of FBAC, we deployed a video-on-demand system on the real network as an experimental platform for algorithm evaluation.

5.1. Setup

In most cases in this paper, N video sessions were connected to the video server located on the Internet through the same network bottleneck link. The receiver of each video session is the DASH client running on Google Chrome, and the sender is the modified Nginx-quiche video server in the Alibaba Cloud ecs.s6 instance.

The video dataset used in this paper contains 8 different types of 4-min 4K videos, each of which was re-encoded at 6 different bitrate levels and split into 4-s video chunks. Each video block was pre-computed with its corresponding VMAF scores. We used the VMAF score for the calculation of QoE utility.

5.1.1. Evaluation Indicators

The quality of experience of all users is calculated as shown in Equation (1), where the VMAF score is within the range [0, 100]. Furthermore, in this paper, we aim to improve the QoE fairness among users and maximize the overall QoE efficiency. Therefore, the following two indicators were used to measure the performance of the algorithm:

(1) QoE fairness: Jain's Fairness index was used to measure the QoE fairness in multi-video user scenarios;
(2) Overall QoE efficiency: the QoE mean of all users, namely:

$$\frac{1}{n}\sum_{i\in N} QoE_i \tag{13}$$

5.1.2. Comparison Scheme

(1) MPC: We used MPC as the default ABR to train the algorithm for the fair bandwidth allocation, so MPC was used as the benchmark algorithm for evaluation;

(2) Bola: A fixed heuristic algorithm that uses Lyapunov optimization to select the bitrate taking buffer occupancy only into account. Although it is not directly designed for QoE fairness, it achieves relatively good results;
(3) Pensieve [20]: a relatively advanced adaptive algorithm based on deep reinforcement learning.

5.2. Experimental Results

5.2.1. Fair Scheduling Strategy under Fixed Bandwidth

To illustrate the advantages of FBAC in improving QoE fairness and overall QoE efficiency, we firstly evaluated the performance of the fair scheduling algorithm under the condition of fixed bandwidth. The fixed bandwidth scenarios where the bandwidth capacity of the bottleneck link is 1, 2, 4, and 6 (Mbit/s) were simulated by limiting the downlink bandwidth of the server. Specifically, three video clients of the same type competed on a shared bottleneck link and started playing their randomly selected videos from the video dataset simultaneously. We conducted 20 experiments for each comparison scheme and took the average as the final result.

Figure 3a,b shows the performance of FBAC and the three comparison schemes on QoE fairness and QoE efficiency under different fixed bandwidths. It can be seen that, compared with other algorithms, the improvement in QoE fairness and efficiency of the FBAC algorithm is related to the capacity of the bottleneck link. In a higher-speed network link, the improvement of the FBAC algorithm becomes smaller, and the trend is more evident in the QoE fairness indicator. For example, when the bottleneck link capacity is 1 Mbit/s, the FBAC algorithm improves the QoE fairness of MPC by 9%, while the improvement is only 4% when the link capacity is 6 Mbit/s. This phenomenon occurs because increasing the video bitrate gradually reduces the impact of video content on video quality. When the video bitrate is 0.75 Mbps, the average difference in VMAF scores between videos was 10 points, and when the video bitrate is 2 Mbps, the difference was reduced to 5 points. This also shows that the QoE between users is fairer in a higher rate network link.

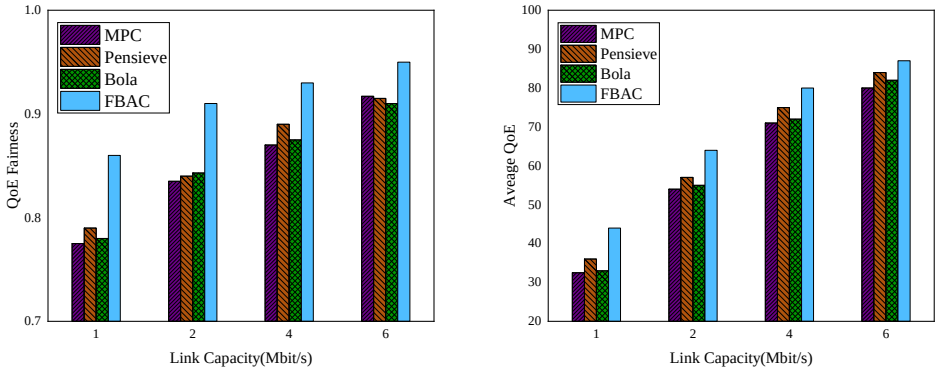
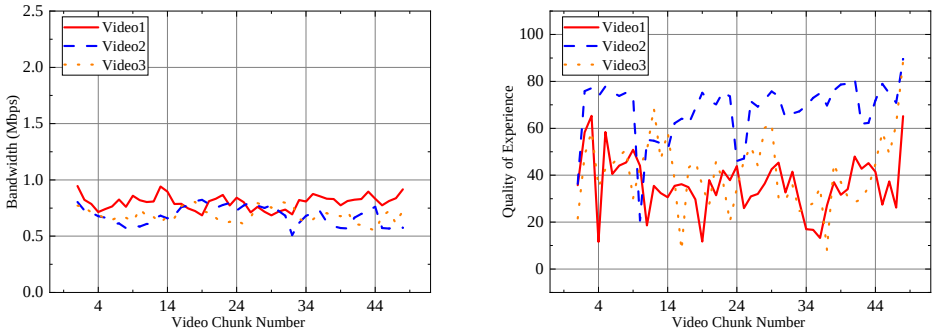


Figure 3. Experimental results of fixed bandwidth. (**a**) QoE fairness; (**b**) QoE efficiency.

In addition, although the increase in available bandwidth also reduced the room for improvement in QoE, when the available bandwidth was 6 Mbps, the overall QoE of FBAC was still 6 points, 3 points, and 4 points higher than that of MPC, Pensieve, and Bola, respectively. When the available bandwidth was 1 Mbit/s, it improved by 8–11 points. To better understand the significance of this improvement, the average difference in VMAF between 720 p and 1080 p video is 7.65 points.

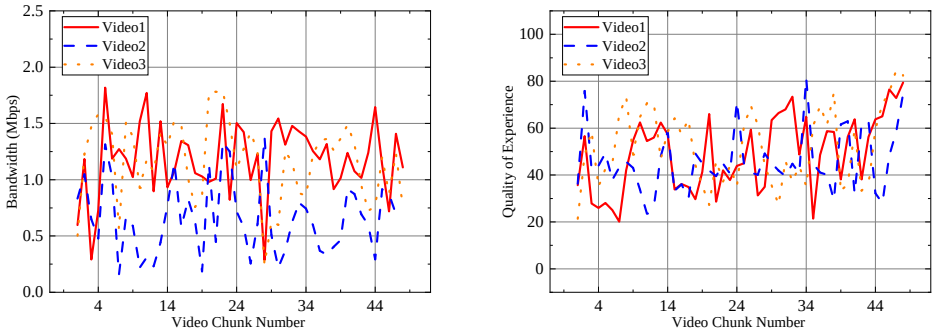
To illustrate how FBAC achieves better QoE fairness, we collect the sending rates and QoE scores of MPC and FBAC under a fixed bandwidth of 2 Mbit/s. As shown in Figure 4, since the underlying congestion control still constrains the ABR algorithm, the bandwidth is fairly allocated to each user, and the difference in video content will lead to unfair QoE among video users with the same available bandwidth. FBAC can perceive the difference

in QoE between users and dynamically adjust the bandwidth allocation between video streams. This indirectly affects the decision of the ABR algorithm, thereby ultimately improving the QoE fairness among the video streams, as shown in Figure 5.



(**a**) (**b**)

Figure 4. (**a**) Evolution of bandwidth in MPC (**b**) Evolution of QoE in MPC.



(**a**) (**b**)

Figure 5. (**a**) Evolution of bandwidth in FBAC (**b**) Evolution of QoE in FBAC.

5.2.2. Fair Scheduling Policy in Dynamic Scenarios

To test the performance of FBAC in dynamic networks, we selected the bandwidth tracking data of real networks to simulate the dynamic changes of bottleneck links to maximize the diversity of network states during the experiment. Considering the randomness of video stream requests in the real network, in this experiment, the three clients participating in the experiment no longer request videos simultaneously but choose random times to randomly order videos in the video dataset.

As shown in Table 1, compared with the other three algorithms, the FBAC algorithm improves fairness by 13%, 11%, and 10%, respectively. The FBAC algorithm focuses on the long-term QoE fairness of video streams, so it has an advantage in dealing with dynamic network environments.

Table 1. Experimental results of under dynamic bandwidth.

Algorithm	QoE Efficient	QoE Fairness
MPC	43.3	0.76
Pensieve	45.1	0.79
Bola	43.9	0.78
FBAC	49.5	0.88

In addition, the average QoE efficiency of FBAC is improved by 13%, 11%, and 9%, respectively, compared with the other three algorithms. To explore how FBAC can improve the overall QoE efficiency, we collected the video quality, re-buffering time, and video smoothness of each scheme in the experimental process. The experimental results are shown in Figure 6, and drew the following conclusions. FBAC can help clients with re-buffering risk to quickly establish buffers, which effectively reduces the re-buffering time of clients and enables video streams to establish a global buffer pool by using shared bottleneck links. This brings another benefit: the clients in the same bottleneck link can obtain higher overall video quality. This is because the perceived quality of the video is a convex function. When bandwidth is allocated from a client playing high-quality video to one playing low-quality video, the experience quality lost by the client playing high-quality video is lower than that gained by the client playing low-quality video.

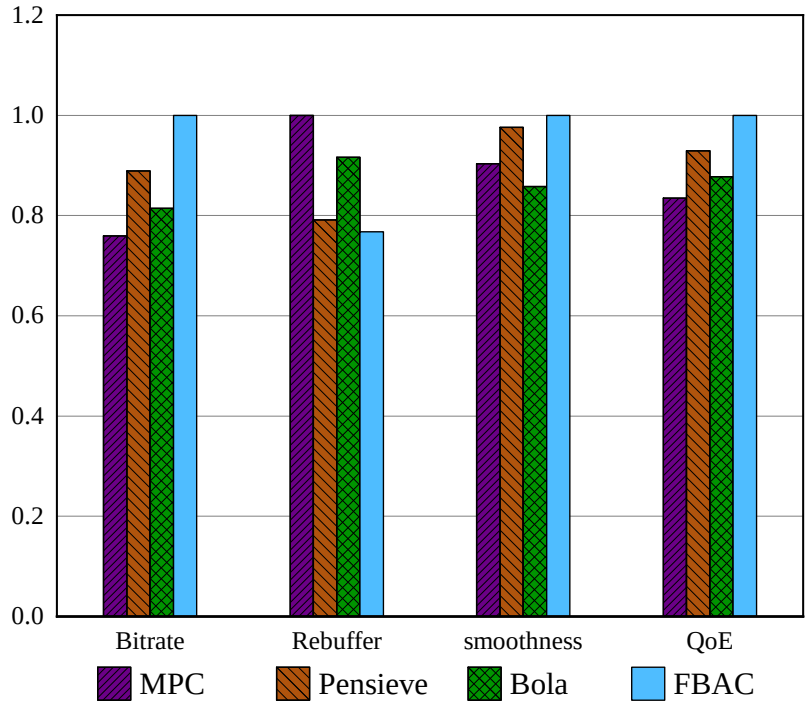


Figure 6. Various indicators of QoE in dynamic bandwidth scenarios.

Furthermore, to verify the impact of global parameters on the distributed algorithm, we trained a version of UN-FBAC that does not use global parameters in the same environment and evaluated it in a dynamic scene. The experimental results are shown in Figure 7. UN-FBAC is inferior to the full version of FBAC in terms of QoE efficiency and QoE fairness. By introducing global parameters as the input parameters of the neural network, the QoE fairness of the FBAC algorithm can be improved by 10%, and the overall QoE efficiency can be improved by 8%.

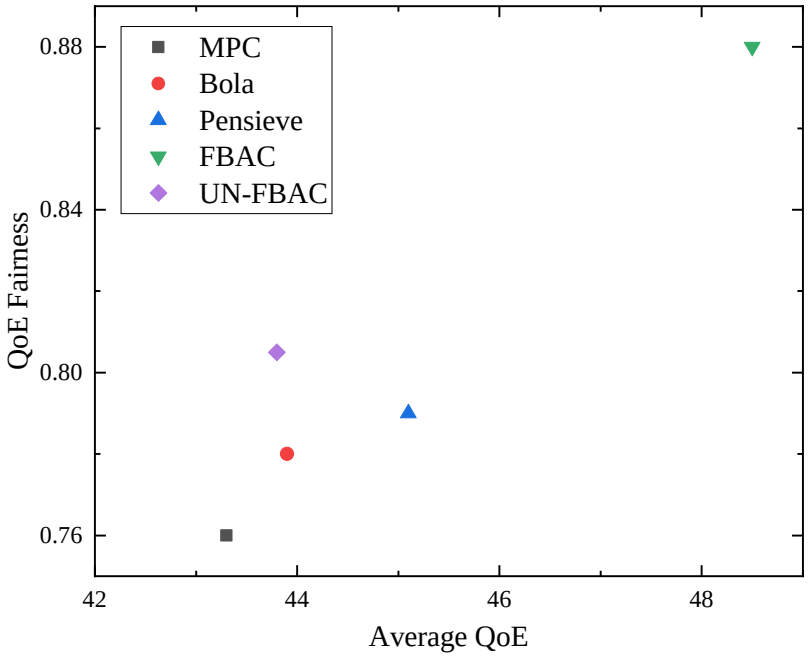


Figure 7. Main experimental results under dynamic bandwidth.

6. Conclusions

In this paper, we designed and implemented a fairness scheduling strategy for multiple video streams based on federated deep reinforcement learning. Based on the previous research, an active bandwidth allocation framework based on congestion control was proposed, enabling the system to allocate bandwidth in a distributed way and ensure the bandwidth utilization of the link. In addition, we also discussed the instability of the environment in the multiple video stream collaboration scenarios. The algorithm's performance was improved by constructing global parameters that contain more information. This experiment showed that the method proposed in this paper is superior to other current solutions.

Author Contributions: Conceptualization, Y.L., W.L., D.W. and C.Z.; methodology, Y.L. and D.W.; software, Y.L., W.L., D.W. and C.Z.; validation, Y.L., W.L. and D.W.; investigation, D.W.; resources, Y.L. and W.L.; writing—original draft preparation, D.W.; writing—review and editing, Y.L., D.W. and C.Z.; supervision, Y.L. and W.L.; project administration, Y.L., W.L., D.W. and C.Z. All authors have read and agreed to the published version of the manuscript.

Funding: Funded by Science and Technology Project of Hebei Education Department ZD2022102.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets generated from this study are available upon reasonable request.

Acknowledgments: The authors gratefully appreciate the anonymous Reviewers for their valuable comments.

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

References

1. Akhshabi, S.; Anantakrishnan, L.; Begen, A.C.; Dovrolis, C. What happens when HTTP adaptive streaming players compete for bandwidth. In Proceedings of the 22nd International Workshop on Network and Operating System Support for Digital Audio and Video, Toronto, ON, Canada, 7–8 June 2012; pp. 9–14.
2. Floyd, S.; Paxson, V. Why we don't know how to simulate the Internet. In Proceedings of the 29th Conference on Winter Simulation, Atlanta, GA, USA, 7–10 December 1997; pp. 1037–1044.

3. Yan, F.Y.; Ayers, H.; Zhu, C. Learning in situ: A randomized experiment in video streaming. In Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20), Santa Clara, CA, USA, 25–27 February 2020; pp. 495–511.
4. Mcmahan, H.B.; Moore, E.; Ramage, D.; Hampson, S. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, Fort Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282.
5. Wang, Y.; Wang, H.; Shang, J.; Hu, T. RESA: A Real-Time Evaluation System for ABR. In Proceedings of the 2019 IEEE International Conference on Multimedia and Expo (ICME), Shanghai, China, 8–12 July 2019; pp. 1846–1851.
6. Spiteri, K.; Urgaonkar, R.; Sitaraman, R.K. BOLA: Near-optimal bitrate adaptation for online videos. In Proceedings of the IEEE INFOCOM 2016—The 35th Annual IEEE International Conference on Computer Communications, San Francisco, CA, USA, 10–14 April 2016; pp. 1698–1711.
7. Yin, X.; Bartulović, M.; Sekar, V.; Sinopoli, B. On the efficiency and fairness of multiplayer HTTP-based adaptive video streaming. In Proceedings of the 2017 American Control Conference (ACC), Seattle, WA, USA, 24–26 May 2017; pp. 4236–4241.
8. Altamimi, S.; Shirmohammadi, S. QoE-fair DASH video streaming using server-side reinforcement learning. *ACM Trans. Multimed. Comput. Commun. Appl.* **2020**, *16*, 1–21. [CrossRef]
9. Nathan, V.; Sivaraman, V.; Addanki, R.; Khani, M.; Alizadeh, M. End-to-end transport for video QoE fairness. In Proceedings of the ACM Special Interest Group on Data Communication, Beijing, China, 19–23 August 2019; pp. 408–423.
10. Jiang, W.; Ning, P.; Zhang, Z.; Hu, J.; Wang, J. Practical Bandwidth Allocation for Video QoE Fairness. In Proceedings of the International Conference on Wireless Algorithms, Systems, and Applications, Nanjing, China, 25–27 June 2021; pp. 523–534.
11. Li, Z.; Aaron, A.; Manohara, M. Toward a practical perceptual video quality metric. *Netflix Tech Blog* **2016**, *6*. Available online: https://medium.com/netflix-techblog/toward-a-practical-perceptual-video-quality-metric-653f208b9652 (accessed on 8 April 2022).
12. Ha, S.; Rhee, I.; Xu, L. CUBIC: A new TCP-friendly high-speed TCP variant. *ACM SIGOPS Oper. Syst. Rev.* **2008**, *42*, 64–74. [CrossRef]
13. Wei, D.X.; Jin, C.; Low, S.H.; Hegde, S. FAST TCP: Motivation, architecture, algorithms, performance. *IEEE/ACM Trans. Netw.* **2006**, *14*, 1246–1259. [CrossRef]
14. Rhee, I.; Xu, L.; Ha, S. *CUBIC for Fast Long-Distance Networks*; Internet Engineering Task Force (IETF): Fremont, CA, USA, 2016.
15. Lowe, R.; Wu, Y.; Tamar, A.; Harb, J. Multi-agent actor-critic for mixed cooperative-competitive environments. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 6382–6393.
16. Rashid, T.; Samvelyan, M.; Witt, C.D.; Farquhar, G.; Foerster, J.; Whiteson, S. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In Proceedings of the 35th International Conference on Machine Learning, Stockholm Sweden, 10–15 July 2018; pp. 4295–4304.
17. Yu, C.; Velu, A.; Vinitsky, E.; Wang, Y.; Wu, Y. The Surprising Effectiveness of MAPPO in Cooperative, Multi-Agent Games. *arXiv* **2021**, arXiv:2103.01955.
18. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.
19. Yin, X.; Jindal, A.; Sekar, V.; Sinopoli, B. A control-theoretic approach for dynamic adaptive video streaming over HTTP. *ACM SIGCOMM Comput. Commun. Rev.* **2015**, *45*, 325–338. [CrossRef]
20. Mao, H.; Netravali, R.; Alizadeh, M. Neural adaptive video streaming with pensieve. In Proceedings of the Conference of the ACM Special Interest Group on Data Communication, Los Angeles, CA, USA, 21–25 August 2017; pp. 197–210.