EAVS: Edge-assisted Adaptive Video Streaming with Fine-grained Serverless Pipelines

Biao Hou*, Song Yang*, Fernando A. Kuipers†, Lei Jiao§, Xiaoming Fu‡
*School of Computer Science and Technology, Beijing Institute of Technology, China
†Delft University of Technology, The Netherlands
§Department of Computer Science, University of Oregon, USA
‡Institute of Computer Science, University of Göttingen, Germany
Email: {houbiao, S.Yang}@bit.edu.cn, F.A.Kuipers@tudelft.nl, jiao@cs.uoregon.edu, fu@cs.uni-goettingen.de

***Abstract*—Recent years have witnessed video streaming gradually evolve into one of the most popular Internet applications. With the rapidly growing personalized demand for real-time video streaming services, maximizing their Quality of Experience (QoE) is a long-standing challenge. The emergence of the serverless computing paradigm has potential to meet this challenge through its fine-grained management and highly parallel computing structures. However, it is still ambiguous how to implement and configure serverless components to optimize video streaming services. In this paper, we propose EAVS, an Edge-assisted Adaptive Video streaming system with Serverless pipelines, which facilitates fine-grained management for multiple concurrent video transmission pipelines. Then, we design a chunk-level optimization scheme to address video bitrate adaptation. We propose a Deep Reinforcement Learning (DRL) algorithm based on Proximal Policy Optimization (PPO) with a trinal-clip mechanism to make bitrate decisions efficiently for better QoE. Finally, we implement the serverless video streaming system prototype and evaluate the performance of EAVS on various real-world network traces. Our results show that EAVS significantly improves QoE and reduces the video stall rate, achieving over 9.1% QoE improvement and 60.2% latency reduction compared to state-of-the-art solutions.**

***Index Terms*—Video streaming, Serverless computing, Deep reinforcement learning, Quality of Experience.**

I. Introduction

The proliferation of 5G offers unprecedented opportunities for video service providers to deliver video streaming to users, boosting the popularity of video applications such as live streams and short videos. As reported by Cisco [1], two-thirds of all video streaming applications will be ultra-high-definition by 2023, which also illustrates the ever-increasing Quality of Experience (QoE) requirements. Even though 5G can increase network bandwidth, the Internet may soon be overwhelmed by massive video streaming traffic, which will affect the users' QoE [2]. Consequently, providing video streaming services to users with satisfactory QoE remains a tremendous challenge for video service providers.

Numerous video service providers, such as Youtube, Hulu, and Netflix, apply the Dynamic Adaptive Streaming over HTTP (DASH) protocol to deliver video streaming through Content Delivery Networks (CDN). In such scenarios, the cloud server splits the video into chunks of equal duration and encodes the video chunks at different bitrate levels. Then, client players run an adaptive bitrate (ABR) algorithm to select the appropriate bitrate for the video chunks [3]. In order to prevent significant losses in revenue caused by QoE degradation, video service providers have to optimize critical components, such as bitrate adaptation strategy [4], [5], buffer management [6], [7], and QoE enhancement scheme [8], [9]. In particular, most existing ABR algorithms for bitrate adaptation try to balance the trade-off between video bitrate and network bandwidth under fluctuating network conditions [10], but only the size and download speed of video chunks are considered. Such coarse-grained bitrate adaptation cannot adapt efficiently to dynamic network environments [11], which may incur frequent video bitrate switching and break playback smoothness. This is because initial chunk bitrate decisions may not be necessarily the optimal choice for downloading chunks later, and once made, such decisions cannot be rolled back. In addition, existing approaches often make bitrate decisions at the client-side according to client conditions locally [12], lacking a global view. Given the large-scale distributed nature of the Internet, individual nodes only observe a partial fragment of the video streaming system, which results in poor long-tail performance [13]. An edge platform can provide video streaming service in close proximity to users. Nevertheless, current edge-based solutions primarily leverage edge caching [6] to reduce video transmission latency. Even though the edge platform has a broader view than end users, these solutions rarely take bitrate adaptation decisions at the edge. Consequently, there is a need for an edge-assisted fine-grained video adaptation scheme to ensure proper QoE for users.

To provide tailor-made services, cloud service providers have proposed the serverless computing paradigm [14], typically in the form of Function-as-a-Service (FaaS) [15]. In the serverless architecture, FaaS handles intuitive event-triggered requests to guarantee inter-linked stateless function availability and elasticity [16]. Serverless computing is an ideal candidate for video streaming applications with high data-level parallelism and intermittent activities [17]. Inspired by recent advances in serverless video analytics [18], [19], we leverage the

Song Yang is the corresponding author.

This work has been partially supported by the National Natural Science Foundation of China (NSFC) under Grant No. 62172038, by the U.S. National Science Foundation under Grants CNS-2047719 and CNS-2225949, and by the EU H2020 COSAFE project (Contract No. 824019) and EU Horizon CODECO project (Contract No. 101092696).

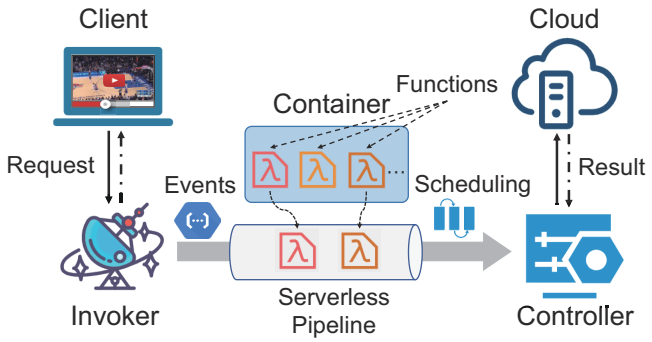
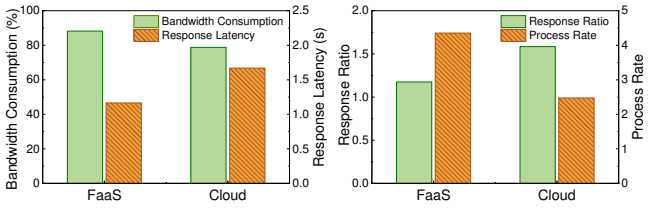


Fig. 1. Architecture of a serverless computing paradigm.



(a) Response latency. (b) Response ratio vs. process rate.

Fig. 2. Performance comparison of FaaS and Cloud for video streaming.

serverless computing paradigm to deploy fine-grained video delivery pipelines at the edge for video streaming applications. To that end, we first propose a high-performance serverless video streaming system enabled by fine-grained video delivery pipelines. Then, we use serverless functions to enhance the response of video request events, thus making bitrate decisions at the edge by leveraging its computation resources. Due to the Markov property of video bitrate adaptation, we devise a Deep Reinforcement Learning (DRL) algorithm based on Proximal Policy Optimization (PPO) [20] together with a trinal-clip mechanism to efficiently address the challenge of bitrate adaptation decisions [21]. Moreover, we incorporate dynamic video chunk quality into the QoE metric to configure the QoE model, by assigning a single priority weight to each video chunk.

To the best of our knowledge, we are the first to propose an edge-assisted adaptive video streaming system with fine-grained serverless pipelines. In particular, this paper makes the following contributions:

- We design a high-performance edge-assisted serverless video streaming system, which uses stateless functions at the edge to make fine-grained bitrate decisions.
- We propose a trinal-clip PPO-based ABR algorithm to boost the robustness of video bitrate decisions for improving QoE during playback.
- We implement a prototype of our edge-assisted serverless video streaming system. Extensive results show that the proposed solution achieves less latency compared with existing approaches, and the proposed PPO-based ABR algorithm outperforms the baselines in terms of QoE.

II. Background and Motivation

This section first presents our implementation setup and the status quo of serverless computing paradigm. Then we use empirical analysis to explain the limitations of existing solutions that underscore our motivation.

A. Implementation Setup

Video streaming system implementation. Based on dash.js [22], an open source DASH player, we implement a video streaming system prototype. Videos from Youtube [23] are used for all experiments, as it is the type of reference videos for dash.js. We deploy an Nginx server based on HTTP/2 to serve the video chunks and use Linux utility TC [24] on the cloud server to control the egress traffic.

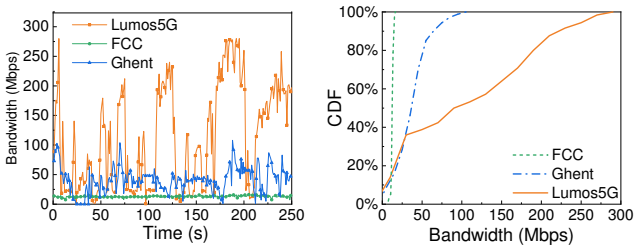
ABR parameters setting. We utilize TensorFlow [25] to train a DRL model. Eight vectors taken as input are transmitted to a convolutional layer (1D-CNN) with 128 filters, and each filter has a size of 4 that is applied with a stride of 1. Then, the results obtained from these layers are aggregated to a hidden layer that applies 128 neurons with ReLu activation function and uses a softmax for the output layer. The critic network has the same neural structure. During the training phase, we set the value of the discount factor $\gamma = 0.95$. Moreover, we set the learning rate to 0.0001 for the actor network and 0.001 for the critic network. The entropy factor β is configured to decay gradually from 1 to 0.1 over 100,000 iterations.

Serverless video pipeline configuration. We implement the primitive of the video streaming pipeline in serverless functions using Python and deploy it on the cloud server and the edge. We configure parallel acceleration to provide high-performance function processing. To deploy the serverless computing architecture at the edge, we use Docker [26] container instances at the edge to execute the serverless function invocations when the invoker receives the chunk request events from clients and interacts with the container environment.

B. Serverless Video Service

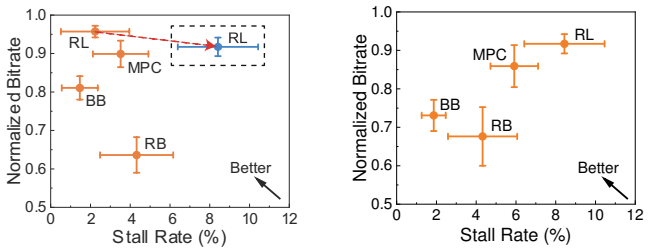
To generate a processing pipeline dedicated to handling specific tasks in serverless computing platforms, developers only need to decouple monolithic applications into multiple independent stateless functions [27]. Fig. 1 shows a serverless computing architecture for a video streaming service. When serverless computing infrastructure is applied to video streaming, the serverless platform is responsible for asynchronous execution of the appropriate stateless function instances with fine-grained response actions in real-time. Once predefined video events invoke, serverless functions concurrently run in response to the occurrence of the corresponding events.

Taking bitrate adaptation function in the video pipeline as an example, we illustrate the advantages of FaaS in video streaming transmission. As shown in Fig. 2(a), FaaS reduces the response latency by 39.4% at only a minimal penalty in bandwidth consumption compared with cloud service. In contrast to plain cloud service, FaaS decouples video streaming



(a) Bandwidth. (b) CDF for bandwidth.

Fig. 3. Bandwidth statistics among different network traces.



(a) QoE impact of 4G trace. (b) QoE impact of 5G trace.

Fig. 4. QoE factors of ABR algorithms under 4G & 5G network conditions.

requests into multiple functions that can execute a number of bitrate decisions of a video file in parallel. It is worth noting that we use fixed bandwidth, while bandwidth is dynamically changing in practice, which may further change the latency value but the decreasing trend still holds. Fig. 2(b) shows the time spent on performing video bitrate adaptation events. FaaS has a 25.8% higher response ratio (which is defined as the ratio of the sum of waiting time and execution time to the execution time) than the cloud service. Meanwhile, the bitrate adaptation processing rate (events per second) of FaaS is higher than the cloud service by 43.2%, achieving rapid and scalable responses. Therefore, we can see that serverless computing can provide the fine-grained response actions needed for service providers to develop video streaming applications.

C. Adaptive Video Streaming

We investigate the influences of ABR algorithms on video streaming by conducting an in-depth investigation of QoE, using prior measurements on commercial networks [28]. We use the 4G datasets FCC [29], Ghent [30] and the 5G dataset Lumos5G [31], respectively, and feed those to the ABR algorithms to study their performance under different network conditions. Overall, we find that some ABR algorithms (e.g., RL-based) can not maintain high performance under 5G networks. Specifically, ABR algorithms suffer from poor performance over 5G for the following reasons:

Sophisticated network conditions. The different network bandwidth traces dynamically fluctuate up to hundreds of Mbps in Fig. 3(a). In addition, we investigate bandwidth distributions on various sets of 4G and 5G network traces, as shown in Fig. 3(b). Some ABR algorithms integrate network throughput into making bitrate decisions. To some extent, their performance depends on the accuracy of the network throughput prediction. The dramatically fluctuating 5G network brings high available throughput while posing a challenge for bitrate adaptation. Therefore, establishing an accurate throughput prediction scheme is crucial for making video bitrate decisions.

Fine-grained bitrate adapation. We examine the performance of four ABR algorithms under different network traces, FCC [29] and Lumos5G [31], as depicted in Fig. 4. Among four ABR optimization algorithms, namely Rate-based (RB) [4], Buffer-based (BB) [5], Model Predictive Control (MPC) [10], and RL [6], we observe a explicit increasing trend in video stall rate by 47.6% on average for RL when running over 5G. In most cases, ABR algorithms under 5G achieve almost similar normalized bitrates in 4G settings at only 2.8% degradation, as shown in Fig. 4(a) and Fig. 4(b). Video stalls are a major problem for video streaming services under 5G. For example, the average video stall for MPC and RL increases by 73.2% and 130.2%, respectively. RL outperforms the other algorithms in 4G but incurs the highest video stall time over 5G networks. A possible explanation is that sometimes the coarse-grained selection of the highest bitrate chunk may lead to high stall times, and it is difficult to alleviate the impact of such wrong decisions on users' QoE. Therefore, we advocate that ABR algorithms should make fine-grained decisions and better adapt to the highly fluctuating network conditions.

III. System Design

This section first introduces the architecture of our serverless video streaming system. Then we discuss core technologies, video streaming with fine-grained serverless pipelines, a PPO-based ABR algorithm, and the QoE metric.

A. System Architecture

We design EAVS, an edge-assisted adaptive video streaming system with serverless pipelines, which facilitates fine-grained management for multiple concurrent video bitrate adaptation events to maximize video QoE. Fig. 5 demonstrates our EAVS architecture, which mainly consists of three parts: clients, edges, and cloud servers. The edge handles video requests from multiple clients and interacts with the geo-distributed cloud servers. Video streaming services can be delivered from the user's nearest cloud server to reduce transmission latency.

At the client side, we extract the JavaScript console logs from the Chrome browser with an integrated dash.js player to collect the state information of the video chunks, player buffer, and network condition using iPerf from the client. Then, the ABR controller organizes state metadata information and forwards it in groups to invoke chunk downloading request events. The corresponding video request invocations are sent to the edge using HTTP via serverless functions.

At the edge side, the functions database is the intelligent core of the whole system and interacts with the container that executes serverless functions. The throughput monitor module probes the network link between the client and cloud

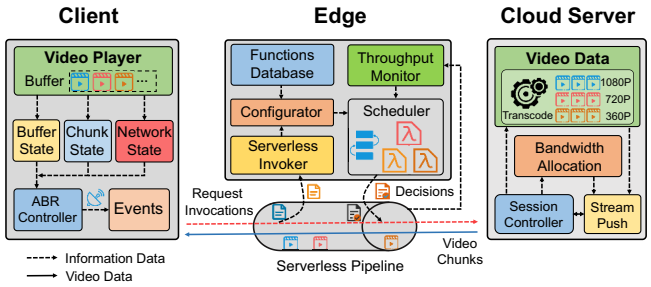


Fig. 5. An overview of our serverless video streaming system.

servers for estimating future throughput. In response to client video request events, the edge decrypts the request information immediately. Specifically, the edge utilizes the information from request events (e.g., buffer occupation and chunk bitrate) and throughput prediction to determine the optimal bitrate using serverless functions. Then the edge delivers the bitrate decision results to the nearest cloud server. Since the edge has a broader view than the client, it can effectively improve video QoE by considering more relevant information, such as client requests and available bandwidth. The serverless video streaming pipelines have an additional streaming ingestion stateless function, which is responsible for orchestrating the execution of the video streaming workflow. During each timeslot, a serverless configurator invokes a stateless function for video streaming events and feeds back pipeline messages. The edge instructs invokers to execute stateless functions for making fine-grained bitrate decisions.

At the server side, each of the cloud servers stores the entire video content and encodes video chunks at different bitrate levels. Once the cloud server receives the video bitrate decision results from the edge through the session controller, it immediately allocates the available bandwidth and determines the optimal streaming push strategy. Finally, the cloud server delivers appropriate video chunks to the client player on the allocated bandwidth using the DASH protocol.

B. Serverless Video Streaming Pipeline

Serverless pipelines provide the off-the-shelf infrastructure to handle fine-grained concurrent video streaming services in real time. By dynamically adjusting the serverless function according to the edge available resources, we aim to: (1) achieve high throughput for video streaming services by serverless pipelines, and (2) adjust the workloads of dynamic video delivery adaptively. To efficiently integrate resources from the edge and cloud server, we deploy an edge-assisted adaptive video streaming system to automatically tune the serverless pipelines. The primary responsibility of the edge is to invoke serverless functions execution and deliver execution results. As a result, the serverless pipeline design follows the single responsibility principle: each stateless function instance takes on a single task independently. The serverless function instantiates when it is invoked and destroys when it is done. During video streaming sessions, this translates to having one fine-grained bitrate decision function for every video event.

Serverless pipelines. We extend the fine-grained serverless computing architecture to boost video streaming service. In order to efficiently deliver video content with a better QoE, cloud service providers need to decompose the monolithic video streaming workflow into a set of sequential stateless functions. For example, the video streaming pipeline is decomposed into inter-linked stateless functions, to be executed by a containerized runtime environment. Therefore, the entire video streaming system becomes serverless pipelines consisting of stateless functions. Operations are not reprofiled when serverless pipeline composites and video requests change. To handle the unstable serverless environment, we propose a video streaming system with event-driven multi-process support [32]. For example, the arrival of video requests invokes serverless functions to make bitrate decisions for downloading chunks. The above process repeats function invocations asynchronously until the video streaming session terminates.

Serverless invoker. During the video session, the serverless invoker module enables the system to orchestrate the execution of all function invocations and maintains the serverless pipeline state, ensuring serverless functions' performance, availability, and responsiveness. The invoker uses a semaphore-based mechanism to control request access to the container. Whenever triggering requests arrive at the edge, it responds by invoking serverless function instances on time and records the invocations statistics in detail, which are used to update the profile required for video service. In addition, our system aims to reduce the overall pipeline latency while minimizing the cost (resource usage). Therefore, we optimize the overall pipeline execution by iteratively and dynamically optimizing the invocation of each operation using up-to-date information about the state of video streaming requests and resource availability. Additionally, we set different latency slack values for each function invocation. We dynamically reduce the system latency by continuously configuring function operations for serverless pipelines subject to meeting a slack value at minimal cost. Through dynamic slack target assignment, pipeline latency can be reduced without considering possible conditional paths. Finally, the serverless video pipeline information is packaged and sent to the configurator.

Configurator. To meet the latency target for the serverless video pipelines, the configurator module identifies two key factors: (1) how much latency slack to set for each function invocation, and (2) how to allocate resources efficiently to meet the latency target. The volatility of independent function execution processes and system resources requires the configurator to dynamically determine the most efficient resource allocation for each function invocation. The configurator continuously detects available resources (e.g., memory and CPU) on the serverless backend. In addition, the configurator module opens a connection to the functions database and searches for available serverless functions to select the invoked operation. After a configuration option is selected, the configurator forwards the function invocation with the configuration decision profiles to the scheduler for execution. We use this latency mechanism to improve the performance of the serverless functions.

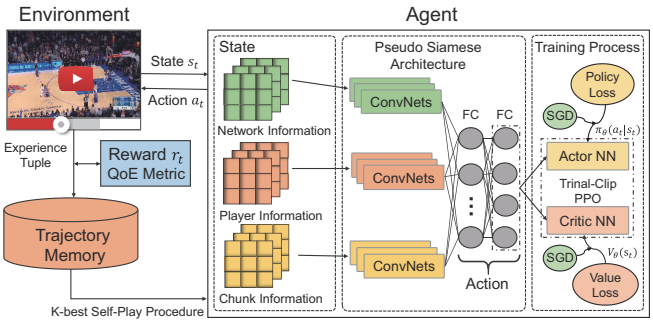


Fig. 6. End-to-end DRL architecture of trinal-clip PPO-based ABR algorithm.

Scheduler. After the configurator module completes a function invocation configuration profile, the invocations are dispatched to the scheduler module for execution. The scheduler performs the serverless function invocation on the container environment specified by the configurator. During serverless function execution, operation invocation may execute asynchronously and concurrently. The scheduler keeps track of execution time for function invocations. If the configuration latency target is exceeded or the scheduler receives an execution error, the scheduler notifies the serverless invoker to recreate a function invocation. This duplicate invocation is then passed to the configurator to reset the slack allocation and execute the configuration process. We set latency targets for different video request invocations to improve pipeline responsiveness, thus ensuring that the function invocation runs in the optimal configuration. When function invocation executes successfully, the scheduler provides the function output results to the cloud server. After processing the input workload, function instances automatically terminate and release the occupied resources.

C. PPO-based ABR Algorithm

We design an end-to-end reinforcement learning framework that performs bitrate adaptation decisions for video streaming. End-to-end means that the framework directly accepts information from video streaming services and outputs actions without the need to encode hand-crafted features as input or iterative reasoning in the decision process. In the DRL-based video streaming scheme, ABR state-space information includes current and historical video streaming information, network information, and action information. The agent chooses the video chunk bitrate based on the current state-space information and tries to win more rewards. To capture the complex relationship among the video streaming state, desired actions, and QoE rewards, we design a pseudo-siamese architecture to learn the intrinsic relationship.

Reinforcement learning (RL) is a classic strategy for solving sequential decision problems in dynamic environments. Deep neural networks (NNs) can also handle multi-dimensional input conditions to extract features in-depth. Consequently, the adaptive streaming bitrate decision problem is within this realm and can be solved possibly by DRL-based solutions. We transform this problem into a Markov Decision Process (MDP) task and devise a DRL-based solution through the interactions between agent and environment. An MDP consists of a set of finite states $\mathcal{S} = \{s_1, s_2, ..., s_t\}$, a set of actions $\mathcal{A} = \{a_t\}_{t=1}^{T}$, and a reward function $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{R}$. After taking action $a_t \in \mathcal{A}$ at each state $s_t \in \mathcal{S}$, the agent will acquire a new state s_{t+1} according to the transition probability strategy and receive a reward $r\left(s_{t+1}|s_t, a_t\right)$. The agent's goal is to maximize the cumulative reward $\mathcal{R} = \sum_{t=1}^{\infty} \gamma^{t-1} r\left(s_{t+1}|s_t, a_t\right)$, where $\gamma \in (0, 1]$ is the discount factor.

As shown in Fig. 6, at each timestamp *T*, the input of the DRL architecture comprises state representations of video chunks and network information, which are sent to the top and bottom streams of the conjoined architecture, respectively. Since actions and state representations provide different kinds of information for learning, we isolate the parameter sharing of the conjoined architecture so that these three ConvNets can learn adaptive feature representations, which are then fused through the fully connected layer to produce the desired actions. The agent can choose an action based on the output probability distribution. To train the DRL model, we resort to a trinal-clip loss function that updates the model parameters using Stochastic Gradient Descent (SGD). We obtain the final model through a self-playing procedure that plays the current model with a pool of *K* best historical versions, so as to sample different training data from a huge state space.

State representation. Existing works typically classify video and network information into distinct groups and use their tandem encoding vectors as representations of states to make iterative reasoning processes feasible. However, the abstracted encoding vectors lose important state information and may not capture the complex relationship between the video information and the optimal bitrate decision. To obtain an efficient and appropriate feature representation that learns directly from the state to the desired bitrate decision, *we design a new multi-dimensional feature representation to encode current and historical video chunks and action information.*

In the video streaming system, the video information and the action information exhibit different characteristics. Therefore, we represent them as two separated three-dimensional tensors and let the neural network learn the system model empirically. We design the tensor as nine channels representing the agent network information, chunk information, player information, and historical decisions. Each channel is a $4 * 13$ sparse binary matrix with each position representing the corresponding information. For the actor tensor, since there are usually at most four consecutive actions per round, we design it in four channels. Each channel is a $1 * n_b$ sparse binary matrix where n_b represents the different bitrates of the video chunk.

Action component. Fig. 6 depicts the overall architecture of the PPO. For the time-series input type (e.g., past chunk download time, etc.), we use a ConvNets layer to extract the basic features efficiently. Then, all processing results are concatenated into a fully connected (FC) layer to learn the intrinsic relationships of the complex features. The output results are finally converted into a softmax layer to calculate the probability distribution of the actions. By adjusting the

parameter θ of the neural network, the policy π_θ can be optimized. So when downloading video chunks, the agent chooses the action that can maximize the long-term cumulative QoE reward during a video streaming session.

Trinal-clip PPO. With a multidimensional feature representation, the pivot factor in policy-gradient training is a deep learning paradigm with a suitable loss function. We employ an actor-critic framework with on-policy training. The agent directly learns a parameterized policy π_θ based on the gradient of the expected return concerning the policy parameter θ. The actor-critic paradigm trains a value function $\mathcal{V}_\theta(s_t)$ and a policy $\pi_\theta(a_t|s_t)$ and updates them iteratively by sampling from the replayed experiences.

We use the vanilla Proximal Policy Optimization (PPO) algorithm [20] to achieve policy improvement via gradient-based parameter updates. The PPO defines the ratio function $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta'}(a_t|s_t)}$ as the probability ratio between the current policy π_θ and the old policy $\pi_{\theta'}$. The advantage function $\hat{A}_t$ ($t \in [0, T]$) describes the improvement between two consecutive states s_{t+1}, s_t, over selecting an action a_t according to current policy π_θ, and the policy loss function $\mathcal{L}_t^p$ is:

$$\mathcal{L}_t^p(\theta) = \hat{\mathbb{E}}_t\left[min\left(r_t(\theta)\hat{A}_t, clip\left(r_t(\theta), 1-\varepsilon, 1+\varepsilon\right)\hat{A}_t\right)\right] \quad (1)$$

where $clip\left(r_t(\theta), 1-\varepsilon, 1+\varepsilon\right)$ ensures that $r_t(\theta)$ lies in the interval $(1-\varepsilon, 1+\varepsilon)$, and ε denotes the clipping range with typical value 0.2 [33]. The value loss $\mathcal{L}_t^v$ is defined as:

$$\mathcal{L}_t^v(\theta) = \hat{\mathbb{E}}_t\left[\left(R_t^\gamma - V_\theta(s_t)\right)^2\right] \quad (2)$$

in which R_t^γ represents the traditional γ-return. We use the min function to control the original and truncated items, preventing the policy update drift beyond a predefined interval.

We apply the optimization method with advantage function estimation and additional entropy rewards to improve the PPO algorithm performance. Moreover, using Generalized Advantage Estimation (GAE) [33] to construct the advantage function can reduce the variance so that the algorithm does not produce large fluctuations. By following the T-step update method TD error ∂_t, we calculate GAE as:

$$\hat{A}_t = \partial_t + (\omega\lambda)\partial_{t+1} + ... + (\omega\lambda)^{T-t+1}\partial_{T-1} \quad (3)$$

where

$$\partial_t = \mathcal{R}_t + \omega\mathcal{V}_\theta(s_{t+1}) - \mathcal{V}_\theta(s_t) \quad (4)$$

However, above PPO loss function has difficulty to converge during large-scale distributed training. We analyze two main reasons for this problem: (1) the policy loss $\mathcal{L}_t^p(\theta)$ introduces unbounded variance when $\pi_\theta(a_t|s_t) \gg \pi_{\theta'}(a_t|s_t)$ and the advantage function $\hat{A}_t < 0$. (2) the value loss $\mathcal{L}_t^\vartheta(\theta)$ is often extremely large due to the uncertainty of the bitrate distribution in video streaming. To speed up and stabilize the training process, we implement a trinal-clip PPO loss function. It introduces an additional clipping hyper-parameter σ_1 for the policy loss and two clipping hyper-parameters σ_2 and σ_3 for the value loss when $\hat{A}_t < 0$. The policy loss function $\mathcal{L}_t^{tcp}$ of the trinal-clip method [34] is defined as:

$$\mathcal{L}_t^{tcp}(\theta) = \hat{\mathbb{E}}_t\left[min\left(clip\left(r_t(\theta), clip\left(r_t(\theta), 1-\varepsilon, 1+\varepsilon\right), \sigma_1\right)\hat{A}_t\right)\right] \quad (5)$$

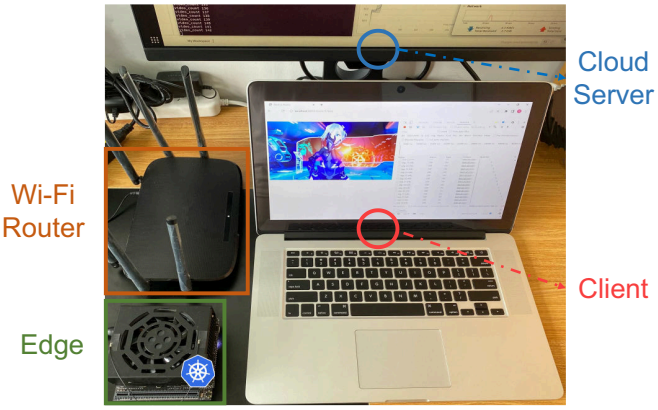


Fig. 7. Implementation of our serverless video streaming system prototype.

where $\sigma_1 > 1+\varepsilon$ indicates the upper bound, and ε denotes the original clip range in the PPO algorithm. The clipped value loss function $\mathcal{L}_t^{tcv}$ for trinal-clip PPO is defined as:

$$\mathcal{L}_t^{tcv}(\theta) = \hat{\mathbb{E}}_t\left[\left(clip\left(R_t^\gamma, -\sigma_2, \sigma_3\right) - \mathcal{V}_\theta(s_t)\right)^2\right] \quad (6)$$

In the DRL training process, the hyper-parameters σ_2 and σ_3 represent the total number of video chunks the player has downloaded and the server has stored, respectively. Thus, these two hyper-parameters do not require manual tuning but are dynamically calculated according to the chunks played during video playback. We implement a distributed version of the trinal-clip PPO algorithm to accelerate the training speed and enhance the training performance. This stringent restriction significantly reduces the variance of the value function and also eliminates the influence of policy irrationality.

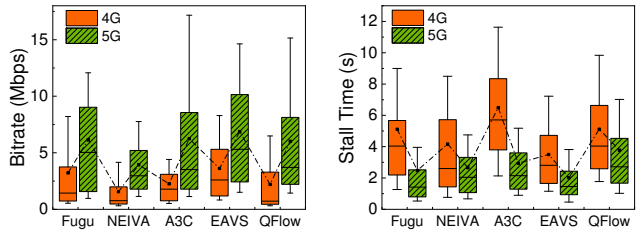
QoE metric. A QoE metric is used as the DRL reward to train our trinal-clip PPO-based ABR algorithm using our proposed system under different network conditions. We use two PCs as video server and client, respectively. The edge is a Jetson Agx Xavier to perform the bitrate decisions. Fig. 7 shows our serverless video streaming prototype implemented by different devices. We reuse existing QoE metrics [21] and assign a weight for each video chunk according to its inherent video content. Hence, we assign a particular weight for each video chunk to encode content quality sensitivity. We perform a video chunk weighting of the QoE metric as follows:

$$\mathbb{R}_i = \sum_{i=1}^{n}\omega_i R_i \quad (7)$$

where ω_i denotes the weight of $chunk_i$ and n is the number of video chunks during the video streaming session, reflecting the priority of this chunk. The DRL model is encouraged to download video chunks at higher bitrates by using a QoE reward function. The adopted QoE metric in our solution is:

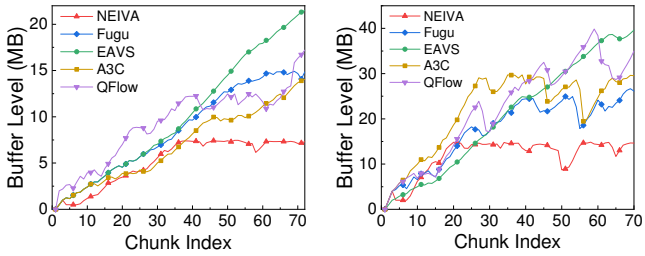
$$QoE = \sum_{i=1}^{n} q(\mathbb{R}_i) - \mu\sum_{i=1}^{i} T_i - \delta\sum_{i=1}^{n-1}\left|q(\mathbb{R}_{i+1}) - q(\mathbb{R}_i)\right| \quad (8)$$

where $\mathbb{R}_i$ is the bitrate of $chunk_i$, $q(\cdot)$ is a utility value which represents the video quality, T_i denotes the video stall penalty,



(a) Bitrate of video chunks. (b) Stall time.

Fig. 8. The performance of different methods in various network traces.



(a) Buffer variations for 4G trace. (b) Buffer variations for 5G trace.

Fig. 9. The buffer level of different methods in various network traces.

and $|q(\mathbb{R}_{i+1}) - q(\mathbb{R}_i)|$ represents the smoothness penalty when switching bitrates. In Eq. (8), μ and δ are non-negative weights, which are set to 4.3 and 1.0, following existing works [2] [21]. In a nutshell, QoE increases with high bitrates, but diminishes with stall-time, and lack of smoothness.

IV. Performance Evaluation

A. Dataset and Methodology

Network trace datasets. We consider different 4G and 5G network traces to evaluate the performance of serverless video streaming under real-world network conditions. In particular, we set up ABR experiments in the serverless pipeline-enabled video streaming scenario. The parameter settings are the same as for Section II-A. We select two public network datasets FCC [29], and Lumos5G [31] which represent the 4G and 5G network conditions, respectively. All traces are formatted by the Mahimahi tool [35]. We randomly split the network dataset into three groups: training, validation and test, with a share of 80%, 10%, and 10% of the dataset.

Video datasets. We evaluate videos from Youtube [23]. For each video, we split the video into fixed-length chunks and encode them at five different bitrate levels (300, 750, 1200, 1850, 2850) Kbps using the FFmpeg [36] tool, corresponding to (240, 360, 480, 720, 1080) P levels.

Baselines. We compare the following state-of-the-art baselines to demonstrate the performance of our proposed solution. All algorithms are implemented in the same environment:

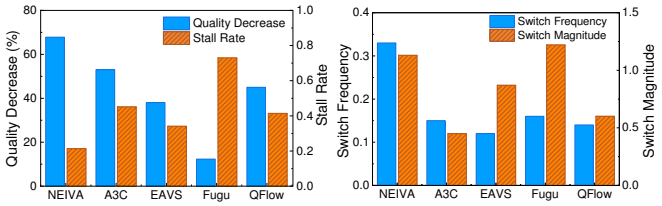
- Fugu [2]: It replaces the network throughput predictor based on a classical controller with a deep neural network of upcoming chunk transmission time, designed to be feasibly trained using supervised learning in situ.
- A3C [6]: It adopts an asynchronous advantage actor-critic (A3C) method, one of the DRL-based algorithms, to solve the corresponding MDP issue of bitrate decisions, by jointly considering video transcoding and edge caching.
- NEIVA [10]: It improves network throughput prediction accuracy by training the predictor with the Hidden Markov Model (HMM), then incorporates Model Predictive Control (MPC) for mobile video streaming.
- QFlow [37]: It develops a model-free Double Deep Q Network (DDQN) that enables bitrate decision adaptation to video feedback control loop to maximize QoE.

B. QoE Metrics Analysis

To understand the QoE metrics obtained by the ABR algorithms, we carefully investigate the meta-metrics implemented for each video player in different network conditions. We demonstrate the performance of the ABR algorithms under different network traces in Fig. 8. Apparently, the 5G traces (green) provide higher throughput than the 4G network traces (red) and hence a 53.8% higher average video chunk bitrate, as shown in Fig. 8(a). Our proposed EAVS performs best, with a video chunk bitrate improvement of about 9.1%-42.7% compared to the other benchmark algorithms. Fig. 8(b) shows the stall time of different ABR algorithms under network traces. We observe that the stall time of EAVS is about 19.1% and 15.6% lower compared to A3C and Fugu, and 11.3% and 8.7% lower compared to NEIVA and QFlow. Through fine-grained chunk bitrate adaptation decisions, EAVS is able to consciously adapt to dynamic network conditions, with higher video bitrates and smaller stall time.

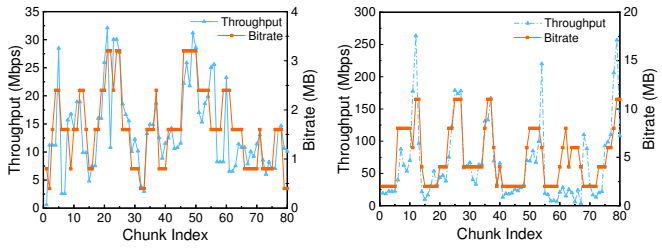
Fig. 9 illustrates buffer occupancy of one identical video (140s) across various network traces. We notice that there are significant differences in buffer management by different algorithms. As shown in Fig. 9(a) and Fig. 9(b), the buffer occupancy drastically fluctuates under 5G networks compared with 4G networks. ABR strategies try to maximize the bitrate of video chunks, but coarse-grained bitrate management often leads to significant variances. EAVS carefully explores the bitrate improvement space within the bandwidth constraint through fine-grained control. Overall, the buffer footprint gradually flattens during the process of downloading chunks.

We only show the QoE results for the 5G network in Fig. 10 and Fig. 12. Fig. 10 presents the QoE performance for a fixed buffer among different ABR algorithms. The change in QoE is shown as a percentage difference in quality and an absolute increase in stall rate. As shown in Fig. 10(a), NEIVA, QFlow and A3C, face a nearly 55% decrease in video quality combined with a slight rise in stall rate. Fugu maintains high quality (less than 20% decrease in quality) but does so at the expense of a significant increase in stall rate. EAVS has less than 40% decrease in quality, but with a lower stall rate than Fugu. Fig. 10(b) demonstrates the frequency and magnitude of bitrate switching over the entire video streaming session. NEIVA leads to the most frequent



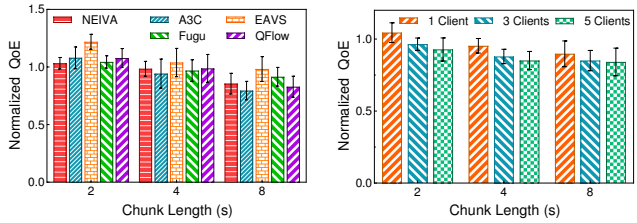
(a) Video quality v.s. stall rate. (b) Video switch events.

Fig. 10. Video QoE performance of different methods on fixed buffer.



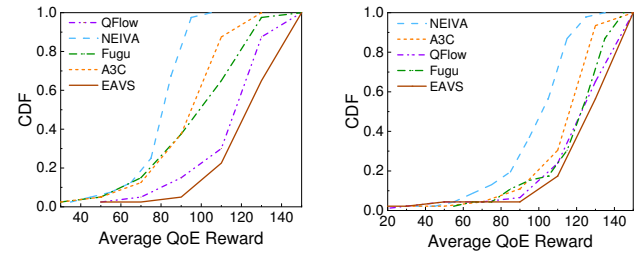
(a) PPO for 4G trace. (b) PPO for 5G trace.

Fig. 11. A case of bandwidth and chunk bitrate by different network traces.



(a) Various methods. (b) Number of clients.

Fig. 12. QoE performance with different chunk length in Dash.js.



(a) Diverse methods for 4G trace. (b) Diverse methods for 5G trace.

Fig. 13. CDF of average QoE reward by various algorithms.

quality switching. The reason is that buffer occupancy levels fluctuate in the presence of video rebuffering, resulting in more bitrate switching. The frequent network fluctuations result in an inability to accurately estimate throughput, which in turn causes a negative drift in video content performance. In contrast, EAVS tends to have a shorter stall by adapting to dynamic network conditions in real-time, which inevitably leads to some video quality degradation. A better trade-off between video bitrate and stall time would improve QoE.

The video chunk bitrate positively correlates with the available throughput, as shown in Fig. 11. We observe that network throughput affects the video chunk size when the bandwidth varies drastically. In Fig. 11(a), from the 25th chunk the bandwidth suddenly decreases from 30Mbps to 3.6Mbps and remains at decline for the following nine video chunks, thus ensuring smooth playback. However, due to the lack of rapid feedback on the bandwidth, EAVS raises the bitrate in three consecutive video chunks from the 61st chunk, which may lead to stall-time, as shown in Fig. 11(b). The average network throughput of the 5G trace is higher than the average bitrate of the video. Under this network trace, EAVS always chooses a higher bitrate, but it can still cause stall-time. Due to the flexibility of fine-grained bitrate decisions, EAVS can efficiently guarantee to get out of tight corners and provide better video streaming service.

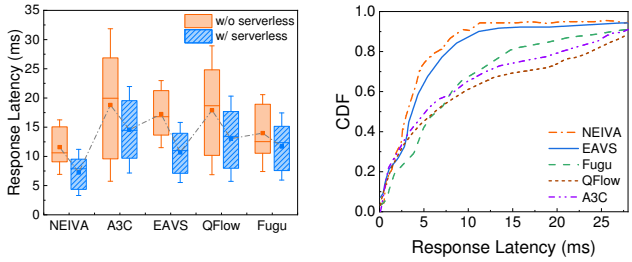
C. End-to-end QoE Improvement

We further validate the robustness of our proposed EAVS solution. To investigate QoE improvement, we show the QoE performance for different video chunk lengths over a 5G network in Fig. 12. We find that longer video chunk sizes cannot enhance the QoE metric defined in Eq. (8), but rather diminish QoE due to the download time. As shown in Fig. 12(a), EAVS experiences a 29.8% QoE degradation. Nevertheless, the QoE of EAVS achieves 4.7%-9.1% improvement, compared to the average QoE of other baselines. As shown in Fig. 12(b), the QoE degradation with an increasing number of clients does not exceed 12% when using EAVS to optimize bitrate decisions. It is limited by the performance of edge devices, which cannot support a large number of serverless functions.

Fig. 13 illustrates the CDF of QoE improvement for various ABR optimization schemes under different network conditions. Fig. 13(a) and Fig. 13(b) display the average QoE reward under 4G and 5G networks, respectively. Compared to the other baselines, we note that EAVS performs well in terms of QoE reward. For example, compared to 4G, the QoE of EAVS improves by 36.4% under 5G network conditions. In addition, the raw values of QoE show that EAVS is the only ABR algorithm that ends up with the maximum average QoE reward. This indicates that EAVS can adapt to underlying dynamic networks faster to improve QoE, because it enables fine-grained serverless service.

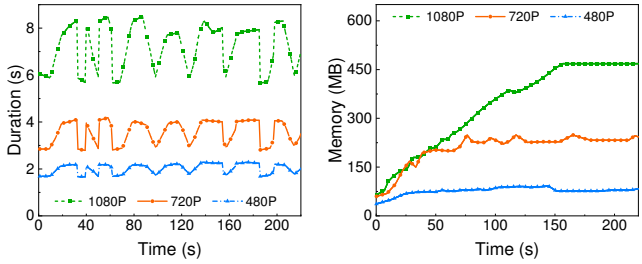
D. Serverless Pipelines Performance

We finally compare these ABR algorithms with and without serverless pipelines at the edge. We demonstrate, for the 5G network trace, the comprehensive performance of serverless video streaming pipelines in Fig. 14. With the edge-assisted serverless video pipelines, EAVS significantly reduces the average response latency by up to 27.4%-60.2% compared to without a serverless pipeline scheme, as shown in Fig. 14(a). It indicates that the serverless configurations can perform well on video bitrate adaptation, and EAVS can pick up an appropriate function for video delivery workloads. Fig. 14(b) shows the



(a) Response latency. (b) CDF for response latency.

Fig. 14. Overall performance among serverless pipelines.



(a) Memory footprint. (b) Process duration.

Fig. 15. Changes in execution time for the encoding function.

CDF of five algorithms under 5G network conditions, where 80% of the response latency of EAVS is less than 10ms. By executing serverless functions in parallel, EAVS can schedule serverless pipelines in response to bitrate decision tasks. This means that our proposed EAVS solution is more effective in reducing the response latency for dynamic changes of incoming workloads in the serverless runtime environment.

We plot the variation of video serverless functions (encoding function) during the video playback in Fig. 15. We observe that in Fig. 15(a) the serverless function execution duration varies increasingly as the video definition increases. For example, the execution duration of the longest 1080P (6,675ms) is 3.7 times higher than the shortest execution duration (1,924ms), and the increment is even bigger than the average execution duration of 720P (3,789ms). In addition, higher definition encoding functions require higher memory space and gradually stabilize during video encoding. As shown in Fig. 15(b), 1080P video is 41.8% higher than 720P and 78.5% higher than 480P, respectively. EAVS gradually indicates a more stable execution duration regardless of the memory footprint. Considering that EAVS invokes concurrently serverless functions for video bitrate adaptation decisions, it may cause competition for resources with other co-located function instances. As a result, the underlying infrastructure is primarily responsible for the apparent performance changes over time.

V. Related Work

Serverless computing. The emergence of serverless computing has extensively simplified deploying applications on the cloud. In the serverless computing infrastructure, functions are orchestrated logically to construct applications. It enables applications to run in a serverless-native manner, including data processing [16] and video analytics [17], [18]. For example, Wukong [15], a new serverless parallel computing framework, provides decentralized scheduling and supports large-scale data processing workloads. Additionally, serverless computing also gains popularity in video scenarios. Zhang *et al.* [19] integrate edge and cloud resources to unleash the potential of serverless computing in enabling scalable real-time video analytics. Llama [38] presents a heterogeneous serverless video processing framework that executes general video pipelines. Konstantoudakis *et al.* [39] propose a serverless immersive media streaming framework that executes video transcoding profiles in a centralized manner. The works closest to us are [19], [38], which leverage the serverless computing technique to decouple video analytic workflow. Unlike these systems, our work is designed for video streaming by decoupling monolithic video streaming applications into independent serverless functions. We further devise a trinal-clip PPO-based ABR algorithm to select the appropriate video bitrate at the edge for QoE improvement.

Adaptive bitrate (ABR) algorithms. ABR algorithms are widely used to select bitrates adaptively according to network conditions. There are two traditional optimization categories: (i) parametric methods that leverage the video information and throughput estimation to make bitrate decisions, such as rate-based algorithms [4], [8], buffer-based algorithms [5], [7], [40], and control theoretic approaches [2], [9], [10]. However, their reliance on accurate throughput estimations limits their performance enhancement. (ii) Learning-based methods that train DRL models to learn bitrate adaptation procedures to make bitrate decisions [41], such as QFlow [37]. However, DRL models in this context are inefficient to train online and hardly converge in highly dynamic environments [42]. Compared with the existing DRL-based ABR algorithms, our proposed trinal-clip PPO-based ABR algorithm can improve sampling efficiency and convergence speed. Moreover, Luo *et al.* [6] present a DRL-based approach to enhance video streaming performance by jointly considering edge caching, playout buffer dynamics and network conditions. Our work is orthogonal to their work [6], [40], because EAVS mainly focuses on making bitrate adaptation decisions via edge-assisted fine-grained serverless pipelines together with a trinal-clip PPO-based ABR algorithm to maximize QoE.

VI. Conclusion

In this paper, we have proposed EAVS, an edge-assisted adaptive video streaming system with fine-grained serverless pipelines. By fully unleashing edge capabilities to facilitate fine-grained serverless management, EAVS can provide good QoE for video streaming services. Moreover, we have devised a trinal-clip PPO-based DRL algorithm to efficiently make bitrate adaptation decisions. Extensive results with our serverless video-streaming prototype have shown that EAVS effectively reduces the average response latency by 60.2% and improves QoE by 9.1%, compared to state-of-the-art solutions.

REFERENCES

[1] Cisco, "Cisco annual internet report (2018–2023) white paper," *Cisco: San Jose, CA, USA*, 2020.

[2] F. Y. Yan, H. Ayers, C. Zhu, S. Fouladi, J. Hong, K. Zhang, P. Levis, and K. Winstein, "Learning in situ: a randomized experiment in video streaming," in *NSDI*, 2020, pp. 495–511.

[3] B. Wang, M. Xu, F. Ren, and J. Wu, "Improving robustness of DASH against unpredictable network variations," *IEEE Transactions on Multimedia*, vol. 24, pp. 323–337, 2021.

[4] Y. Sun, X. Yin, J. Jiang, V. Sekar, F. Lin, N. Wang, T. Liu, and B. Sinopoli, "CS2P: Improving video bitrate selection and adaptation with data-driven throughput prediction," in *Proc. of the ACM SIGCOMM Conference*, 2016, pp. 272–285.

[5] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "BOLA: Near-optimal bitrate adaptation for online videos," *IEEE/ACM Transactions on Networking*, vol. 28, no. 4, pp. 1698–1711, 2020.

[6] J. Luo, F. R. Yu, Q. Chen, and L. Tang, "Adaptive video streaming with edge caching and video transcoding over software-defined mobile networks: A deep reinforcement learning approach," *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 1577–1592, 2020.

[7] T. Huang, C. Zhou, R.-X. Zhang, C. Wu, X. Yao, and L. Sun, "Stick: A harmonious fusion of buffer-based and learning-based approach for adaptive streaming," in *IEEE INFOCOM*, 2020, pp. 1967–1976.

[8] A. Narayanan, X. Zhang, R. Zhu, A. Hassan, S. Jin, X. Zhu, X. Zhang, D. Rybkin, Z. Yang, Z. M. Mao *et al.*, "A variegated look at 5G in the wild: performance, power, and QoE implications," in *Proc. of the ACM SIGCOMM Conference*, 2021, pp. 610–625.

[9] X. Zhang, Y. Ou, S. Sen, and J. Jiang, "SENSEI: Aligning video streaming quality with dynamic user sensitivity," in *NSDI*, 2021, pp. 303–320.

[10] C. Qiao, G. Li, Q. Ma, J. Wang, and Y. Liu, "Trace-driven optimization on bitrate adaptation for mobile video streaming," *IEEE Transactions on Mobile Computing*, vol. 21, no. 06, pp. 2243–2256, 2022.

[11] T. Feng, H. Sun, Q. Qi, J. Wang, and J. Liao, "Vabis: Video adaptation bitrate system for time-critical live streaming," *IEEE Transactions on Multimedia*, vol. 22, no. 11, pp. 2963–2976, 2020.

[12] Z. Meng, Y. Guo, Y. Shen, J. Chen, C. Zhou, M. Wang, J. Zhang, M. Xu, C. Sun, and H. Hu, "Practically deploying heavyweight adaptive bitrate algorithms with teacher-student learning," *IEEE/ACM Transactions on Networking*, vol. 29, no. 2, pp. 723–736, 2021.

[13] F. Y. Yan, J. Ma, G. D. Hill, D. Raghavan, R. S. Wahby, P. Levis, and K. Winstein, "Pantheon: the training ground for internet congestion-control research," in *USENIX ATC*, 2018, pp. 731–743.

[14] M. S. Aslanpour, A. N. Toosi, C. Cicconetti, B. Javadi, P. Sbarski, D. Taibi, M. Assuncao, S. S. Gill, R. Gaire, and S. Dustdar, "Serverless edge computing: vision and challenges," in *Australasian Computer Science Week Multiconference*, 2021, pp. 1–10.

[15] B. Carver, J. Zhang, A. Wang, A. Anwar, P. Wu, and Y. Cheng, "Wukong: A scalable and locality-enhanced framework for serverless parallel computing," in *Proc. of ACM SoCC*, 2020, pp. 1–15.

[16] A. Agache, M. Brooker, A. Iordache, A. Liguori, R. Neugebauer, P. Piwonka, and D.-M. Popa, "Firecracker: Lightweight virtualization for serverless applications," in *NSDI*, 2020, pp. 419–434.

[17] M. Zhang, F. Wang, Y. Zhu, J. Liu, and B. Li, "Serverless empowered video analytics for ubiquitous networked cameras," *IEEE Network*, vol. 35, no. 6, pp. 186–193, 2021.

[18] J. Jiang, S. Gan, Y. Liu, F. Wang, G. Alonso, A. Klimovic, A. Singla, W. Wu, and C. Zhang, "Towards demystifying serverless machine learning training," in *Proc. of the International Conference on Management of Data*, 2021, pp. 857–871.

[19] M. Zhang, F. Wang, Y. Zhu, J. Liu, and Z. Wang, "Towards cloud-edge collaborative online video analytics with fine-grained serverless pipelines," in *Proc. of the 12th ACM Multimedia Systems Conference*, 2021, pp. 80–93.

[20] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv:1707.06347*, 2017.

[21] S. Sengupta, N. Ganguly, S. Chakraborty, and P. De, "HotDASH: Hotspot aware adaptive video streaming using deep reinforcement learning," in *IEEE ICNP*, 2018, pp. 165–175.

[22] Dash.js, https://github.com/Dash-Industry-Forum/dash.js.

[23] "NBA playoffs 2021: Best moments to remember," https://www.youtube.com/watch?v=zw3TIOESmVg.

[24] TC, http://lartc.org/lartc.html.

[25] TensorFlow, https://www.tensorflow.org/.

[26] Docker, https://www.docker.com/.

[27] C. Lin and H. Khazaei, "Modeling and optimization of performance and cost of serverless applications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 3, pp. 615–632, 2020.

[28] X. Yang, H. Lin, Z. Li, F. Qian, X. Li, Z. He, X. Wu, X. Wang, Y. Liu, Z. Liao *et al.*, "Mobile access bandwidth in practice: Measurement, analysis, and implications," in *Proc. of the ACM SIGCOMM Conference*, 2022, pp. 114–128.

[29] FCC, "Measuring broadband raw data releases - fixed," https://www.fcc.gov/oet/mba/raw-data-releases, 2021.

[30] J. Van Der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck, "HTTP/2-based adaptive streaming of HEVC video over 4G/LTE networks," *IEEE Communications Letters*, vol. 20, no. 11, pp. 2177–2180, 2016.

[31] A. Narayanan, E. Ramadan, R. Mehta, X. Hu, Q. Liu, R. A. Fezeu, U. K. Dayalan, S. Verma, P. Ji, T. Li *et al.*, "Lumos5G: Mapping and predicting commercial mmWave 5G throughput," in *Proc. of the ACM IMC*, 2020, pp. 176–193.

[32] V. Mittal, S. Qi, R. Bhattacharya, X. Lyu, J. Li, S. G. Kulkarni, D. Li, J. Hwang, K. Ramakrishnan, and T. Wood, "Mu: an efficient, fair and responsive serverless framework for resource-constrained edge clouds," in *Proc. of the ACM SoCC*, 2021, pp. 168–181.

[33] D. Ye, Z. Liu, M. Sun, B. Shi, P. Zhao, H. Wu, H. Yu, S. Yang, X. Wu, Q. Guo *et al.*, "Mastering complex control in moba games with deep reinforcement learning," in *Proc. of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 6672–6679.

[34] E. Zhao, R. Yan, J. Li, K. Li, and J. Xing, "Alphaholdem: High-performance artificial intelligence for heads-up no-limit poker via end-to-end reinforcement learning," in *Proc. of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 4, 2022, pp. 4689–4697.

[35] R. Netravali, A. Sivaraman, S. Das, A. Goyal, K. Winstein, J. Mickens, and H. Balakrishnan, "Mahimahi: Accurate record-and-replay for HTTP," in *USENIX ATC*, 2015, pp. 417–429.

[36] FFmpeg, https://ffmpeg.org/.

[37] R. Bhattacharyya, A. Bura, D. Rengarajan, M. Rumuly, B. Xia, S. Shakkottai, D. Kalathil, R. K. Mok, and A. Dhamdhere, "QFlow: A learning approach to high QoE video streaming at the wireless edge," *IEEE/ACM Transactions on Networking*, vol. 30, no. 1, pp. 32–46, 2022.

[38] F. Romero, M. Zhao, N. J. Yadwadkar, and C. Kozyrakis, "Llama: A heterogeneous & serverless framework for auto-tuning video analytics pipelines," in *Proc. of ACM SoCC*, 2021, pp. 1–17.

[39] K. Konstantoudakis, D. Breitgand, A. Doumanoglou, N. Zioulis, A. Weit, K. Christaki, P. Drakoulis, E. Christakis, D. Zarpalas, and P. Daras, "Serverless streaming for emerging media: towards 5G network-driven cost optimization," *Multimedia Tools and Applications*, vol. 81, no. 9, pp. 12 211–12 250, 2022.

[40] F. Wang, F. Wang, J. Liu, R. Shea, and L. Sun, "Intelligent video caching at network edge: A multi-agent deep reinforcement learning approach," in *IEEE INFOCOM*, 2020, pp. 2499–2508.

[41] H. Yeo, Y. Jung, J. Kim, J. Shin, and D. Han, "Neural adaptive content-aware internet video delivery," in *OSDI*, 2018, pp. 645–661.

[42] X. Zuo, J. Yang, M. Wang, and Y. Cui, "Adaptive bitrate with user-level QoE preference for video streaming," in *IEEE INFOCOM*, 2022, pp. 1279–1288.