Mamba: Bringing Multi-Dimensional ABR to WebRTC

Yueheng Li*
Nanjing University
Nanjing, China
yueheng.li@smail.nju.edu.cn

Zicheng Zhang*
Nanjing University
Nanjing, China
zichengzhang@smail.nju.edu.cn

Hao Chen†
Nanjing University
Nanjing, China
chenhao1210@nju.edu.cn

Zhan Ma
Nanjing University
Nanjing, China
mazhan@nju.edu.cn

ABSTRACT

Contemporary real-time video communication systems, such as WebRTC, use an adaptive bitrate (ABR) algorithm to assure high-quality and low-delay services, e.g., promptly adjusting video bitrate according to the instantaneous network bandwidth. However, target bitrate decisions in the network and bitrate control in the codec are typically incoordinated and simply ignoring the effect of inappropriate resolution and frame rate settings also leads to compromised results in bitrate control, thus devastatingly deteriorating the quality of experience (QoE). To tackle these challenges, Mamba, an end-to-end multi-dimensional ABR algorithm is proposed, which utilizes multi-agent reinforcement learning (MARL) to maximize the user's QoE by adaptively and collaboratively adjusting encoding factors including the quantization parameters (QP), resolution, and frame rate based on observed states such as network conditions and video complexity information in a video conferencing system. We also introduce curriculum learning to improve the training efficiency of MARL. Both the in-lab and real-world evaluation results demonstrate the remarkable efficacy of Mamba.

CCS CONCEPTS

• **Information systems** → **Multimedia streaming**.

KEYWORDS

Adaptive bitrate (ABR), multi-dimensional ABR, QoE, multi-agent reinforcement learning

*Both authors contributed equally to this research.
†Corresponding author: Hao Chen.

1 INTRODUCTION

The exponential surge of real-time video communication (RTVC) applications, such as video conferencing, remote sharing, and online education, has been observed in recent years. With the emergence of COVID-19, these applications have even become a part of our daily lives, resulting in notable consumption of internet video traffic (e.g., exceeding 17% [3]), and significant revenue returns (e.g., billions of dollars worldwide [21]).

In video-on-demand (VoD) applications, content providers commonly encode the videos at different bitrate levels in advance and resort to adaptive bitrate (ABR) technology to combat the underlying network fluctuations for quality optimization [14, 31]. However, in the RTVC applications, bitrate adaption becomes more challenging due to the unavailable actual compressed video bitrate. The ABR strategy implemented in WebRTC [28], a typical RTVC framework, encompasses two primary processes: congestion control (or bandwidth estimation/target bitrate decision[1]) and video bitrate control. The former involves estimating the available bandwidth by analyzing network state metrics such as packet loss rate and round-trip time (RTT) interval. The estimated bandwidth then serves as the target bitrate for the application-layer encoder to encode a video. To make the video bitrate match the target bitrate, the encoder adaptively determines the quantization parameter (QP) for each frame using the built-in rate control algorithms [29]. Meanwhile, WebRTC dynamically adjusts the resolution or frame rate to compensate for the inaccurate bitrate control of the encoder.

However, such a paradigm suffers from several serious problems, making the performance of existing RTVC applications far from optimal. Firstly, the incoordination of the transport layer and application layer weakens the effectiveness of the ABR strategy (see Figure 1). This is brought about by the limited information sharing and mismatch of decision granularity between the congestion control and bitrate control [5, 11, 34]. Moreover, existing frame rate or resolution adaptation strategies use preset rules in the form of fixed heuristics, which cannot adapt to diverse video content (see Figure 2). Finally, the unsophisticated combination of the rate control algorithm in the encoder and external adaptation policy for frame rate and resolution fails to achieve a perfect collaboration, consequently degrading the final performance (see Figure 3). Recently, reinforcement learning (RL)-based congestion control methods [2, 33, 34] have emerged with the aim to improve

---

[1]In this paper, we use these three terms interchangeably to refer to the same algorithm.

Yueheng Li, Zicheng Zhang, Hao Chen, and Zhan Ma

the accuracy of bandwidth estimation at the transport layer. Nevertheless, they do not step out of the aforementioned paradigm and thus share the same drawbacks.

This paper proposes a novel end-to-end multi-dimensional adaptive bitrate (MABR) framework for RTVC applications, named Mamba, which considers both network conditions and video content complexity to decide the encoding factors (e.g., QP, frame rate, and resolution) that jointly determine the video bitrate. The proposed Mamba avoids the use of explicit target bitrate, which may lead to incoordination between the transport and application layers. Instead, it directly optimizes the user's QoE in an end-to-end manner based on network conditions and video content complexity information. To this aim, we model the joint adaption of the encoding factors as a multi-agent decision problem and devise a multi-agent reinforcement learning (MARL) approach to learn the appropriate combination of the encoding factors through exploring the environment. During the training, curriculum learning is introduced to improve the training efficiency and asymptotic performance (model performance after reaching convergence) of Mamba.

In summary, our contributions are as follows:

- Based on deep analysis of the ABR strategy used by WebRTC (§2.1), we identify the limitations of widely-used WebRTC on providing a satisfactory QoE (§2.2) and further propose an end-to-end multi-dimensional ABR framework to overcome these limitations (§3).
- To the best of our knowledge, this is the first work that employs multi-agent reinforcement learning to adaptively adjust the joint encoding factors including QP, frame rate, and resolution, to optimize the QoE of RTVC applications based on observed network conditions and video content complexity (§4).
- We have integrated Mamba into the WebRTC framework and evaluated its performance by comparing it to state-of-the-art ABR algorithms. Extensive experimental results in real-world field tests indicate that Mamba outperforms the native WebRTC with improvements in video quality of 16.9% while maintaining a similar frame rate and delay (§5).

2 BACKGROUND AND MOTIVATION

Nowadays, RTVC has emerged as the principal medium for individuals to engage in social, recreational, and professional activities. As the first open-source and consistently updated RTVC framework, WebRTC [28] has gained significant traction and has been supported natively by various popular browsers such as Chrome, Firefox, and Safari. The technologies employed by WebRTC, e.g., congestion control, adaptive bitrate, and packet loss recovery, have been adopted in various other RTVC applications, including low-latency live streaming, cloud gaming, and video conferencing [2, 10]. This work presents a novel ABR framework for the optimization of existing WebRTC systems. Nonetheless, the proposed ABR framework holds potential for broader applicability across all RTVC applications.

2.1 Adaptive Bitrate in WebRTC

The ABR policy in WebRTC can be broadly divided into two primary cascaded processes: target bitrate decision and bitrate control. In the following, we provide a concise summary of these two parts.

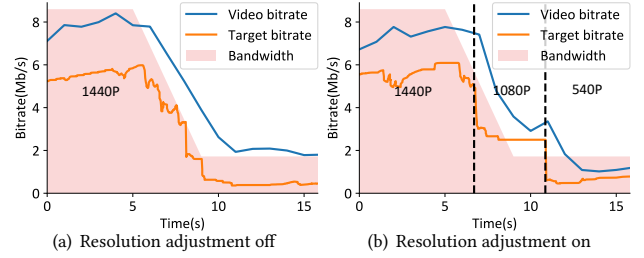(a) Resolution adjustment off   (b) Resolution adjustment on

Figure 1: The mismatch between target bitrate and video bitrate using WebRTC in both resolution adjustment on and off modes, reflects the incoordination issue between transport and application layers.

Target bitrate decision. WebRTC adopts the Google Congestion Control (GCC) [6] algorithm to estimate current network capacity and further employs the estimated bandwidth as the target bitrate for video encoding. The GCC algorithm comprises two controllers: the packet loss-based controller and the delay-based controller. The former undertakes three different actions on the sending bitrate based on the packet loss rates found in RTCP packets: decrease, maintain, or increase. Meanwhile, the latter first filters round trip time (RTT) variations to divide the network into three states: overuse, underuse, and normal, and then performs corresponding decrease, increase, or maintenance operations on the sending bitrate accordingly. The final sending bitrate is determined by the minimum value of the two controllers' outcomes.

Bitrate control. Initially, WebRTC uses the estimated network bandwidth generated by GCC as the target encoding bitrate, at which the video encoder is called for video compression using its built-in rate control algorithm (e.g., in variable bitrate (VBR) or constant bitrate (CBR) mode). However, it is realized by adjusting the QP of each frame only, without considering the resolution and frame rate, which also determine the final video bitrate as reported in [13]. Therefore, WebRTC also integrates a resolution adjustment module and a frame rate adjustment module to further enhance its bitrate control. The resolution adjustment module builds a fixed mapping table with separate entries for all bitrate ranges and corresponding resolution configurations. By looking up this table, the resolution is determined for a given target encoding bitrate. On the other hand, the frame rate adjustment module employs a leaky bucket algorithm to skip the next frame(s) whenever the actual video bitrate surpasses the target bitrate. It is worth noting that for the current version of WebRTC, the resolution adjustment module and the frame rate adjustment module cannot be activated concurrently, and the former is enabled by default. In this paper, we keep the default settings in the implementation of WebRTC.

2.2 Observations

In this section, we set up preliminary experiments to evaluate the ABR strategy of WebRTC and yield a set of remarkable findings. These experiments were conducted on our testbed which is detailed in §5.1.

2.2.1 Incoordination between transport and application layers. As introduced in §2.1, WebRTC's ABR policy is composed of two separate algorithms (i.e., target bitrate decision and bitrate control)
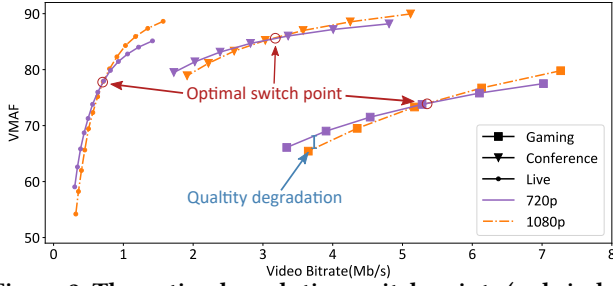
Figure 2: The optimal resolution switch points (red circles) vary for different types of videos. A sub-optimal switch could lead to a noticeable quality degradation (blue mark).

which are deployed at the transport layer and the application layer respectively, and short of shared information except for the target encoding bitrate. This results in coordination issues, including adaptation lag and optimization objective misalignment, as observed in previous works [11, 34]. To demonstrate these issues, we conducted an experiment by simulating network congestion events with the setting of varying available bandwidths and streaming a video that is randomly chosen from the YouTube User Generated Content (YT-UGC [26]) dataset following the WebRTC protocol.

The performances of WebRTC with resolution adjustment on and off, are illustrated in Figure 1. Figure 1(a) reveals a significant discrepancy between the target encoding bitrate and the actual video bitrate for WebRTC, primarily due to the inaccurate bitrate control by the encoder in low-delay encoding mode. The target bitrate generated by GCC decreases rapidly at around 7s, while the video bitrate generated by the encoder fails to follow it closely. This bitrate offset further impairs the accuracy of bandwidth estimation for GCC (after 10s). As a result, prolonged overshooting occurs (after 10s) due to ineffective bitrate control of WebRTC, which seriously deteriorates the user's QoE. Although WebRTC in later versions has introduced a dynamic resolution adaptation scheme to mitigate the incoordination problem, it is still far from satisfactory. As shown in Figure 1(b), the mismatch between target bitrate and video bitrate still exists, and overshooting is observed from 7s to 13s. We desire an end-to-end adaptation policy that straightforwardly performs bitrate control based on network conditions and video information, bypassing inaccurate bandwidth estimation.

2.2.2 Limitation of fixed rules-based resolution adaptation. Even though WebRTC introduces a resolution adaptation policy to enhance the efficiency of bitrate control, it is fixed rules-based, which does not generalize to different video contents. We conducted an experiment to demonstrate this issue by selecting three video contents with different complexity from the YT-UGC dataset: live streaming, video conferencing, and games. These videos were encoded with 1080P and 720P resolutions at different bitrates. VMAF [20] is employed to evaluate picture quality. As depicted in Figure 2, the optimal resolution switching point for different content varies considerably, and a sub-optimal switching point leads to a drop in VMAF by up to 5. This emphasizes the importance and necessity of a more effective resolution adaption strategy.

2.2.3 Importance of joint adaption of QP, resolution, and frame rate. In addition to the first preliminary experiments, we conducted another experiment to show the impact of the above-mentioned
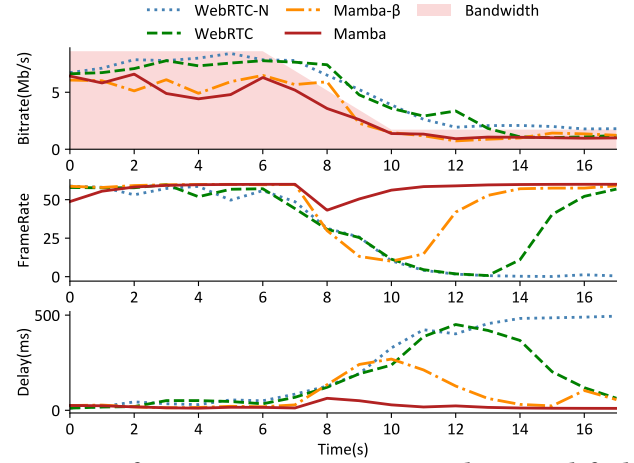

Figure 3: Performance comparison using WebRTC-N, default WebRTC, Mamba-β, and full Mamba. The outstanding performance of Mamba confirms the importance of a joint adaption of QP, resolution, and frame rate.

WebRTC's limitations on QoE metrics, including video bitrate, playback frame rate, and delay. Using the same environment configuration as §2.2.1, we compare the results using WebRTC with resolution adaptation off (WebRTC-N), default WebRTC, Mamba with the joint adaption of QP and resolution (Mamba-β), and full Mamba (with a joint adaptation of QP, resolution, and frame rate). As shown in Figure 3, the video for WebRTC-N almost entirely freezes after 12 seconds, while that for WebRTC eventually resumes normal playback. This again highlights the significance of dynamic resolution adaptation. Mamba-β achieves a faster recovery time and lower delay than WebRTC, which verifies the effectiveness of the end-to-end adaptation policy. Furthermore, Mamba applies an additional frame rate adaptation to reduce the video bitrate more quickly when facing network congestion, avoiding excessive packet losses. By actively downgrading the encoding frame rate, Mamba increases the likelihood that the received video frames can be decoded and achieves a higher playback frame rate and lower delay, resulting in a higher QoE. This finding reveals that a joint adaptation of QP, resolution, and frame rate can further improve ABR performance.

3 MULTI-DIMENSIONAL ABR ALGORITHM

Based on the experimental analysis presented in §2.2, we learn that the existing WebRTC framework cannot ensure a satisfactory QoE due to incoordination between the transport and application layers and the unsophisticated combination of multi-dimensional bitrate control policies. In this paper, we propose Mamba, an end-to-end multi-dimensional ABR framework, designed to learn end-to-end adaptation policies that directly map the observations (e.g., network conditions, and video information) to the best configurations of encoding factors including QP, resolution, and frame rate for QoE optimization. A comparison of the ABR framework between Mamba and WebRTC is illustrated in Figure 4. Both Mamba and WebRTC use a monitor to collect network conditions and video information and finally make decisions for the video encoder to adapt to the network dynamics. However, WebRTC splits the ABR process into two separate sub-processes, i.e., target bitrate decision
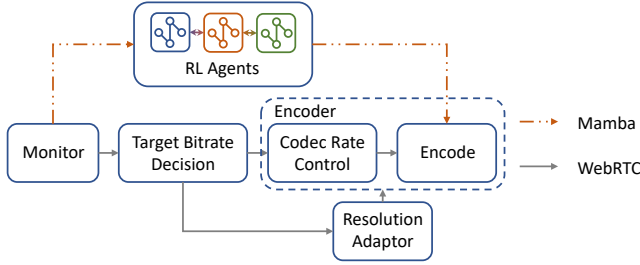
Figure 4: The key brilliance of Mamba compared to default WebRTC.

and bitrate control, and the bitrate control is also fulfilled by two separate modules, i.e., rate control in the encoder and resolution adaptation externally, resulting in the incoordination issues previously discussed. In contrast, Mamba uses a MARL-based model to automatically generate joint ABR algorithms for the end-to-end optimization from observations to the decision of encoding factor configurations, which avoids these incoordination issues and consequently improves the ABR performance. Prior studies have used single-agent reinforcement learning to facilitate the target bitrate decision process [2, 33]. However, the large discrete action space of the MABR task, e.g., 168 (7×4×6) combinations of the atomic action in this work, is difficult for a single agent to handle [16].

To this aim, we model the MABR problem as a multi-agent task that is composed of multiple autonomous agents that interact with the same environment and concurrently influence it. In detail, *qua*, *res*, and *fr* agents are introduced to be in charge of adapting the QP, resolution, and frame rate, respectively. From the perspective of an individual agent, the effects of other agents' actions are integrated into the environment. Consequently, during the training, the environment is non-stationary for each agent as the policies of other agents are updated continuously. To overcome the non-stationary issue, we have meticulously designed the training process of Mamba, more details about which are provided in §4.2. In the following, we describe how we model the MABR problem as a typical MARL task and choose the appropriate algorithm to solve it by explaining some essential concepts in the field of multi-agent tasks.

Local or Global States. The first concern lies in the locality of the states that are used for training. Local states can be observed by the agent in the inference, while global states are only available during the training. In the design of Mamba, the states input to an agent is predominantly local, except for the encoding factors that will be decided by other agents. For example, the resolution and the frame rate applied to next-interval video frames, are two global states for the *qua* agent, since they are being decided by the *res* agent and the *fr* agent respectively, and are not available. Further details regarding the states used by Mamba are introduced in §4.1.

Homogeneous or Heterogeneous Agents. In multi-agent tasks, the homogeneity of learning agents refers to that multiple agents interact with an environment described by the same state space and take actions from the same action space. Homogeneous agents can leverage parameter-sharing techniques to accelerate the training process. However, in the design of Mamba, multiple agents are used to make decisions on different encoding factors, resulting in their disparate action spaces. Considering the discrepancy of decision tasks for each agent, the time span of state observations

also differs, which leads to different state spaces. To address the challenge of training inefficiency resulting from these disparities, this paper proposes the use of curriculum learning, which will be detailed in §4.2.

Cooperative or Competitive Task. A multi-agent task can be cooperative, competitive, or hybrid mode. The classification is based on whether the optimization goals of each agent are fully identical, fully distinct, or partially identical. In the MABR task, the common goal of all agents is to optimize the overall QoE by maximizing the video quality in both spatial and temporal domains and minimizing the end-to-end video delay at the same time. This optimization goal is achieved by using a shared reward function, which is introduced in §4.3. Thus, MABR represents a cooperative task with identical optimization goals for all participating agents.

To sum up, the MABR is a cooperative task involving a group of heterogeneous agents that take actions based on both local and global states. In this paper, we use a state-of-the-art MARL framework, MAPPO [32], to learn an ABR policy for Mamba. MAPPO has been demonstrated to perform well on cooperative tasks and heterogeneous multi-agent problems [19]. To train Mamba's agents, we adopt a centralized training and decentralized execution (CTDE) approach, where the actors make decisions based on their own local states and a shared critic provides a centralized *value* based on global states. In the inference, we only use actor networks to represent Mamba's joint policy, and thus only local states are needed.

4 MAMBA DESIGN

This section first introduces the detailed design of each agent, including the state space, action space, and neural network structure. These designs clarify the responsibilities of each agent and allow them to complete their tasks appropriately (§4.1). We then introduce our carefully designed training methodology to cope with the complex multi-agent explorations, enabling the agents to cooperate with each other to achieve a satisfying QoE (§4.2). Finally, we provide a detailed description of the training algorithm we use (§4.3).

4.1 State, Action and Neural Network Design

For each of Mamba's agents $k \in \{qua, res, fr\}$, they take actions a_t^k based on their local observations including network conditions and video information about the environment as input state s_t^k, according to its policy $\pi_\theta^k(s_t^k, a_t^k)$ which is represented by neural networks with parameters θ.

State. To enable Mamba to coordinate the bandwidth estimation and bitrate control and jointly make decisions on QP, resolution, and frame rate, we include the network conditions, encoder parameters, and video content complexity in the input states. In detail, the state used by Mamba at step t is defined as $s_t = (\vec{u}_t, \vec{v}_t, \vec{c}_t, \vec{y}_t, \vec{f}_t, \vec{h}_t, \vec{d}_t, \vec{p}_t)$, and each of them is a vector of values observed in past K intervals (e.g., $\vec{u}_t = \{u_{t-K+1}, \ldots, u_t\}$). Here we omit the superscript k of the state which indexes the agent because every agent uses the same state variables and only the range of statistical time differs. u_t and v_t respectively represent the average Spatial perceptual Information (SI) and average Temporal perceptual Information (TI) of the raw frames in the same interval counted from step $t-1$ to step t. We include these metrics to measure content complexity which

significantly impacts the rate-distortion [13, 18]. We calculate them following the standard reported in ITU-R BT.1788 [9]. c_t, y_t, and f_t denote the average rate factor value, resolution, and average frame rate, respectively, which are applied to all frames in the interval for encoding. All of them are obtained from the encoder settings. h_t, d_t, and p_t are the playback frame rate, average RTT, and packet loss rate calculated during this interval, respectively. In this work, the interval is set to 0.1s for *qua* and *fr* agents and 1s for *res* agent. The same settings also apply to the action interval for these agents. Specifically, the playback frame rate is defined as the number of actually played frames divided by the interval time. In this work, we set $k = 6$ to capture dynamics of observations in the past 6 intervals, with the time span of the state for both *qua* and *fr* agents as 0.6s (6×0.1s) and that for *res* agent as 6s (6×1s).

Action. The goal of the *qua* agent is to decide the QP for encoding the frames in the next interval. Considering better perceptual quality can be reached by using constant rate factor (CRF) mode than constant QP mode when encoding multiple frames, we use the CRF encoding mode and dynamically adjust the rate factor in this work. In order to avoid dramatic picture quality fluctuations, we build the action set of *qua* agent with incremental adjustments of rate factor, i.e., $a_t^{qua} \in \{+8, +4, +2, 0, -1, -2, -4\}$. Considering the GCC in the default WebRTC framework adopts a strategy that is conservative to improve the video bitrate for congestion avoidance and is more aggressive to decrease the video bitrate when congestion occurs, we set this asymmetric rate factor list to keep this characteristic. For the *fr* agent, we empirically configure the 7-level absolute frame rates as its action set, i.e., $a_t^{fr} \in \{60, 50, 40, 30, 20, 10, 0\}$. Both of the above two agents collect states and perform decisions every 0.1s. The intuition behind this is that the agents need to gather adequate network state information at each decision point while responding promptly to network congestion. For the *res* agent, we pick up four resolution configurations widely used in mainstream RTVC scenarios, i.e., $a_t^{res} \in \{1440p, 1080p, 720p, 540p\}$. The aspect ratio of the video is kept at 16:9. The *res* agent collects states and performs decisions every 1s, as every resolution switch requires a restart of the video encoder.

Neural networks. Mamba utilizes the multi-agent actor-critic framework to generate its control policy. In this framework, separate neural networks are employed for each agent's actor and critic, and a shared critic neural network is additionally introduced to produce a centralized *value* signal. To focus on ABR framework optimization, we design a lightweight neural network architecture inspired by prior works [2, 33], which applies to all actor and critic networks in the multi-agent actor-critic framework. The state inputs are first passed into a gated recurrent unit (GRU) layer with 64 units to capture temporal features, which are then fed into two cascaded fully connected networks with 64 and 32 neurons respectively. All of the hidden layers use the rectified linear unit (ReLU) as the activation function. Finally, the output layer employs the Soft-Max activation function to obtain the final probability distribution of action options.

4.2 Training Methodology

To support joint adaptation of the QP, resolution, and frame rate, Mamba uses MARL to learn an end-to-end multi-dimensional ABR policy. The most intuitive idea is to train Mamba's policy directly
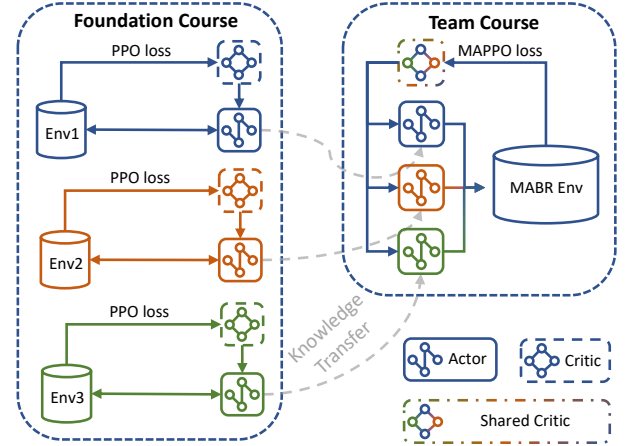

Figure 5: The two-stage curriculum learning that Mamba uses to train its ABR policy.

from scratch using the existing MARL algorithm. However, it may be very inefficient with unacceptable training time, due to the inherent non-stationarity problem of multi-agent tasks, as discussed in §3. As a result, direct training from scratch can easily prevent an agent from exploring the optimal policy since it is unable to cooperate with other agents perfectly [7, 16]. To address the issues, we introduce curriculum learning in the training of Mamba, which has been proven to be helpful in multi-agent training [25, 30]. As shown in Figure 5, we design a two-stage curriculum for MABR learning: foundation course and team course.

Foundation course. During the foundation course, a unique environment is established for each agent. For instance, for the *qua* agent, the environment is transformed from non-stationary to stationary by fixing the resolution and the frame rate and blocking the policies of the other two agents. Consequently, the state inputs involving resolution and frame rate are masked, and the corresponding neurons are frozen in the neural networks during the training. Although the knowledge acquired by the agents in this environment may differ somewhat from the ultimate policy, their semantic experiences remain consistent [25]. In this stage, we train each agent's policy individually using the vanilla PPO algorithm [23], and the training for each agent can be considered as a single-agent reinforcement learning process.

Team course. The objective of the team course is to enable the agent to refine its own learned policy and gain the ability to collaborate with other agents. To achieve this, we initialize the Mamba using the weights obtained from the previous stage, which enables the Mamba to quickly acquire an amount of knowledge about the environment and prevents it from getting stuck in the early stage of training. In this stage, we use the complete state inputs and unfrozen neural networks for training. We utilize the MAPPO algorithm [32] to train Mamba's policy in this stage, and the basic training algorithm will be introduced in the following subsection.

4.3 Basic Training Algorithm

Reward. The goal of Mamba's RL agents is to maximize the expected cumulative discounted reward that they receive from the
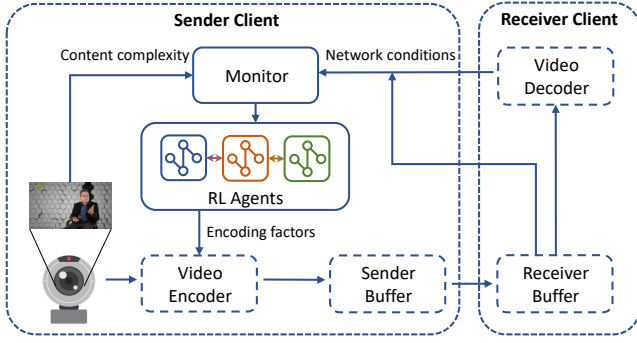
Figure 6: The system implementation of Mamba. We integrate an additional monitor module and the Mamba agents into the existing WebRTC system to build our testbed.

environment. Different from existing works that actually optimize the quality of service at the transport layer (e.g., throughput), we set the reward to directly reflect the user's QoE at the application layer, including the quality of the encoded video, the fluency of video playback, and end-to-end video delay. Specifically, the reward function could be formulated as follows:

$$r_t = \lambda q_t + \nu h_t - \mu d_t. \tag{1}$$

Here, q_t is the perceptual video quality at step t. In order to reflect the picture quality more accurately and directly, we use the rate factor to calculate q_t instead of video bitrate. h_t represents the playback frame rate, which directly affects the fluency of video playback. d_t refers to the average frame delay (in seconds), which is used to approximate the end-to-end video delay between the sender and the receiver in a RTVC session. λ, ν, and μ are weight coefficients that are used to adjust the bias of the policy. In this work, we set them to 1, 8, and 6 empirically. Note that changing these three parameters did not affect the convergence of the model, but had a significant impact on the preference of the final policy.

Loss function. The loss function to train Mamba's policy is defined as:

$$\mathcal{L} = - \sum_t \sum_k \min \left(ratio_{\theta,t}^k, \text{clip} \left(ratio_{\theta,t}^k, 1 - \epsilon, 1 + \epsilon \right) \right) A_t^{GAE}$$
$$+ \beta \sum_t \sum_k H \left(\pi_\theta \left(s_t^k \right) \right), \tag{2}$$

where the first term is the widely-used clipped surrogate loss function, and the second term is the policy entropy which is added to avoid falling into sub-optimal policies at the start of the training. The advantage function A_t^{GAE} is computed using the GAE [22] method. This value measures the difference between the expected reward obtained when the agent picks an action a_t in state s_t deterministically, and the expected reward obtained through actions following the policy π_θ with the policy parameters θ. In this formulation, $ratio_{\theta,t}^k$ and $H(\cdot)$ correspond to the surrogate objective and policy entropy, respectively. The index k identifies a learning agent in Mamba, where $k \in \{qua, res, fr\}$. The value of entropy weight parameter β decreases when the reward no longer improves for 100 epochs. Further information regarding the technical details of the training algorithm is available in [32].

5 EVALUATION

5.1 Methodology

Implementation. We implemented Mamba as an independent module to integrate into a WebRTC-compatible testbed, with the aim to replace the original congestion control and bitrate control modules at the system level. All of the experiments for observation (§2.2) and evaluation were conducted based on this testbed. Figure 6 exhibits the seamless integration of Mamba into the existing WebRTC system. The modules marked by solid lines in the figure are added by Mamba, primarily consisting of Mamba agents and a monitor to gather state observations for Mamba. And the remaining modules indicated by dashed lines inherit from the original WebRTC system. The monitor module collects the observations of content complexity information and network conditions, which are subsequently input into Mamba's RL agents to make decisions on encoding factors for the video encoder. Additionally, we also implemented a state-of-the-art congestion control algorithm Loki [33] in the testbed as a comparison to better evaluate Mamba. In the implementation, we utilized CRF mode for Mamba and VBR mode for WebRTC and Loki (as they can only generate a target bitrate for the video encoder). In the experiment, a sender client running on a PC connects to a receiver client running on another PC via a relay, in which the Linux traffic control tool is used to emulate different network conditions using network traces collected in the real world. At the sender client, a virtual camera is deployed to capture the prepared source videos, which are then processed to generate WebRTC-compatible video streams. This setup provides a controlled environment that enables us to compare the performances of different ABR approaches under the same network condition.

Videos. In order to ascertain the impact of video content on ABR performance, a set of representative videos was chosen from the YouTube UGC video dataset [26], which comprises a diverse range of video genres such as games, video conferences, news interviews, and sports. These videos were concatenated into a continuous 2-minute sequence to create the test video source. This ensures the consistency of the video source for each algorithm during the evaluation process.

Network traces. As with prior studies [2, 33, 34], we conducted controlled laboratory simulations on several public network trace datasets, including: (1) Orca [1]: an AT&T and T-Mobile cellular network trace dataset collected in New York City using mobile devices when the cellular user is walking, stationary, and riding a bus; (2) NYU-METS [15]: a set of LTE mobile network traces collected in New York City MTA on bus and subway riding; (3) Belgium-4G [24]: a 4G network trace dataset collected in Belgium on foot and in a variety of vehicles including bicycles, buses, trams, and trains; (4) US-5G [17]: a set of 5G network traces collected in the US in both SA and NSA modes. Obviously, the considered traces cover diverse access networks, regions, and scenarios, which best capture different network behaviors in the real world. Notably, each trace was post-processed with a granularity of 0.5s for bandwidth sampling. We randomly partitioned these traces into two sets, 80% for training and the remaining 20% for testing.

Evaluation metrics. In the field of RTVC, the user's QoE serves as the most basic metric to evaluate the quality of service. Since the QoE is susceptible to multiple factors from both transport and

Table 1: Comparing Mamba with default WebRTC (GCC) and Loki on Belgium-4G, US-5G, Orca, and NYU-METS network trace datasets.

Dataset	Methods	Video rate (kbps)	VMAF	Delay (ms)	Frame rate (fps)
Orca [1]	WebRTC	4053.02	48.60	124.70	55.56
	Loki	6800.42	50.15	159.48	48.30
	Mamba	4266.06	49.39	122.82	55.98
NYU-METS [15]	WebRTC	4127.83	48.12	144.02	55.05
	Loki	7140.96	48.82	184.06	48.24
	Mamba	4966.93	53.91	137.85	54.79
Belgium-4G [24]	WebRTC	4599.25	50.33	127.59	55.75
	Loki	9279.64	61.51	154.48	50.20
	Mamba	7023.10	60.90	125.34	55.86
US-5G [17]	WebRTC	9180.24	67.97	111.91	55.76
	Loki	18594.67	80.69	129.60	51.36
	Mamba	14294.29	80.44	115.08	55.98
Overall	WebRTC	5490.08	53.76	127.06	55.53
	Loki	**10453.92**	60.29	159.91	49.53
	Mamba	7637.60	**61.16**	**125.27**	**55.65**

application layers, it is challenging to develop a definitive QoE model for assessment. To mitigate this issue, we use several popular QoE metrics to evaluate Mamba's performance from various perspectives, including average video bitrate, average picture quality, average playback frame rate, and average delay. For the picture quality assessment, we use the Video Multi-method Assessment Fusion (VMAF [20]), a widely recognized objective perceptual quality metric, to ensure fair comparisons. Previous studies [8] have reported that VMAF is more closely related to human perception than other metrics such as Peak Signal-to-Noise Ratio (PSNR), Structural Similarity (SSIM [27]), etc. The video bitrate and picture quality are both captured at the sender client, which are used to represent the QoE sensation without considering the network-induced distortions. We use the average round-trip time (RTT) as an indicator to measure the interaction delay experienced throughout the entire session. Ultimately, an optimal ABR policy should prioritize the improvement of picture quality and playback frame rate while minimizing the delay to achieve a satisfactory QoE.

5.2 In-Lab Performance

We evaluated the performance of Mamba and compared it with WebRTC and Loki using the methodology described in §5.1. The average results using all algorithms on different datasets are listed in Table 1. Figure 7 provides more detailed results in the form of cumulative distribution functions (CDF). Based on the analysis of these results, we can yield three key findings.

Firstly, Mamba achieves the best overall performance on almost each QoE metric across different networks. Specifically, Mamba outperforms WebRTC and Loki on the metric of VMAF (by 13.7% and 1.4%), delay (by 1.4% and 21.7%), and playback frame rate (by 0.2% and 12.4%) with an acceptable video bitrate. In contrast, WebRTC tends to be conservative which is evidenced by its lowest video bitrate, leading to the worst picture quality in VMAF. And the

aggressive Loki achieves the highest video bitrate (but not picture quality) at the cost of the largest delay and the lowest playback frame rate, which also fails to provide a satisfactory QoE for users. Furthermore, Mamba achieves consistently better performance under varying network conditions due to its advanced end-to-end multi-dimensional ABR policy. Figure 7 illustrates the absolute advantage of Mamba over other algorithms in the form of CDF.

Secondly, Mamba gains the superior capability of bandwidth adaptation by bridging the incoordination between transport and application layers. This is evidenced by significantly higher video bitrate (which results in a substantial improvement in picture quality) for Mamba compared to WebRTC, while still ensuring slightly better frame rates and delay. This demonstrates Mamba's capability to swiftly recover from network congestion while utilizing as much bandwidth as possible.

Thirdly, Mamba achieves a higher video quality in both spatial and temporal domains even at a relatively lower bitrate. As shown in Table 1, Mamba achieves a higher VMAF and a higher playback frame rate compared to Loki using a lower video bitrate (roughly 3Mbps). The reason is that Mamba learns a joint adaptation policy of QP, resolution, and frame rate, which can dynamically choose an appropriate combination of these encoding factors to achieve desirable video quality.

We would like to emphasize that Mamba can change its policy tendency by adjusting the weights in the reward function (Equation (1)) to suit diverse user preferences or different application platforms. For example, we can train another policy for Mamba using bigger ν and μ and smaller λ, leading to lower delay and higher playback frame rate (i.e., temporal quality) at the cost of lower picture quality (i.e., spatial quality).

5.3 Real-World Performance

To evaluate the generalization capability of Mamba, we conducted a series of real-world experiments covering a wide range of scenarios (stationary and moving) and network conditions (4G and WIFI). The stationary scenarios are comprised of accessing the campus WIFI network under a mixed mode of 2.4GHz and 5GHz within university buildings. Meanwhile, the moving scenarios involve accessing the 4G network while traveling by e-bike at a speed of approximately 20km/h or bus route in a city. All other experimental settings remained the same as described in the preceding section.

The results of the experiments are presented in Figure 8, which demonstrates the superior performance of Mamba compared to WebRTC and Loki. Mamba displayed better performance than WebRTC, with a more timely response to network congestion and efficient bandwidth utilization. This leads to significant improvements in picture quality (about 16.9% improvement in VMAF) while maintaining high frame rates. Loki achieves a similar picture quality as Mamba while suffering significant frame rate degradation (about 4fps), which indicates more stalling. These results confirm the effectiveness of Mamba in RTVC applications.

6 DISCUSSION

Although we implemented and evaluated the proposed Mamba specifically on the WebRTC platform to avoid complicated engineering issues, our method can also be applied to other RTVC
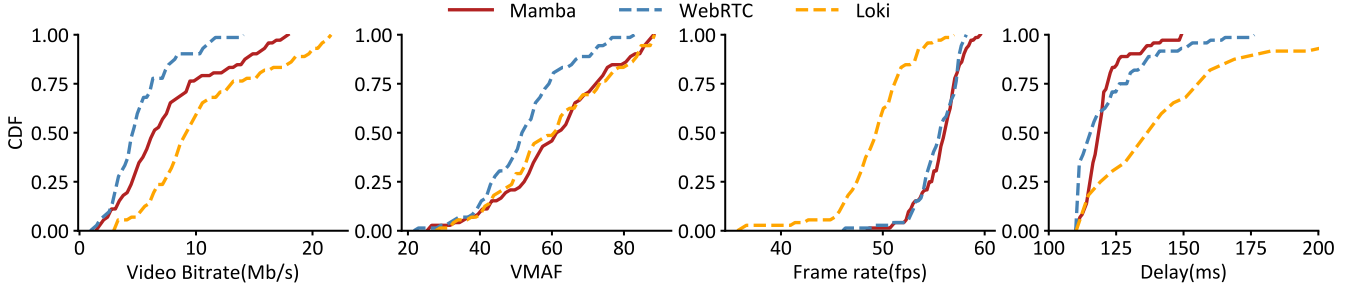
Figure 7: Comparing Mamba with the state-of-the-art WebRTC and Loki in the form of CDF distribution. The metrics of video bitrate, picture quality (in VMAF), playback frame rate, and delay are used to evaluate their performances. Results were collected on all network datasets.

(a) Stationary scene with WIFI network.   (b) Mobility scene with 4G network.
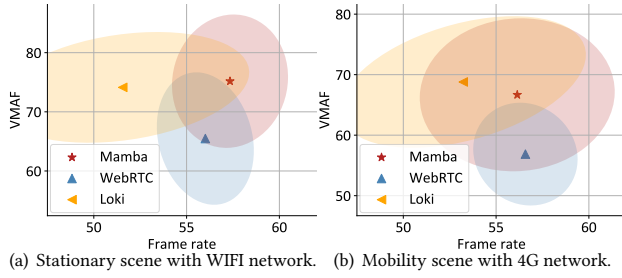
Figure 8: Real-world evaluation results of Mamba, WebRTC, and Loki in both stationary and moving scenarios.

applications which require high picture quality and low interaction delay, such as cloud gaming and virtual reality applications. Therefore, our contribution is not limited to optimizing the WebRTC and video conferencing experience. Furthermore, the MARL training algorithm used in this study is based on off-the-shelf advanced algorithms (e.g., MAPPO [32]) with state-of-the-art performance. Even when more advanced algorithms emerge in the field of MARL, our ABR approach will still be applicable.

7 RELATED WORK

Real-time video communication. WebRTC is a widely used framework that utilizes rules-based GCC for congestion control and separate modules for rate control. However, the incoordination between the transport and application layers seriously degrades the performance. To address this issue, Salsify [5] was proposed to dynamically adjust the frame rate based on bandwidth estimation by accessing the internal state of the encoder. Unfortunately, Salsify's simple hand-crafted strategies and lack of support for resolution adaptation lead to poor performance. Furthermore, Salsify requires a specific encoder, making it difficult to deploy on existing RTVC platforms. Recently, several learning-based methods have been proposed to enhance congestion control performance. T-Gaming [2] used RL to improve congestion control performance in cloud gaming, while Concerto [34] utilized imitation learning (IL) to improve congestion control performance in mobile video telephony. Furthermore, Loki [33] proposed a hybrid method that combines RL and IL to achieve state-of-the-art congestion control performance. However, these learning-based methods still follow

the same paradigm as WebRTC and suffer from the same limitations as previously mentioned.

Multi-agent reinforcement learning. Multi-agent reinforcement learning has been proven to have significant advantages in solving complex problems. Based on the training paradigm, it can be mainly classified into two categories: fully decentralized, and centralized training and decentralized execution (CTDE). IPPO [4] is a representative algorithm of the fully decentralized training methods, which is typically applied in scenarios where each agent can access global observation states and the agents are homogeneous (which can share parameters between each other). MAPPO [32] and MADDPG [12] are representative algorithms of CTDE methods, where each agent usually can only observe local state values and use a shared reward signal for parameter updates. To better solve the problem of non-stationary environments in MARL, CTDE typically uses global state values as input to the critic for better training. The latter's characteristics are more suitable for the MABR task. In addition, to make training more efficient, curriculum learning was proposed in CM3 [30] and DyMA-CL [25] to guide agents by incrementally increasing the difficulty of the agents' tasks, which has also been introduced in this work to improve the training efficiency.

8 CONCLUSION

We introduced and evaluated Mamba, a novel end-to-end multi-dimensional ABR framework for real-time video communication, which leverages multi-agent reinforcement learning to effectively coordinate the target bitrate decision and bitrate control processes and jointly determines the encoding factors including quantization parameters, frame rate, and resolution, resulting in an end-to-end bitrate adaptation that prioritizes QoE considerations. Through both in-lab and real-world tests in diverse scenarios, we demonstrate that Mamba outperforms existing state-of-the-art approaches across a wide range of networks and scenes.

REFERENCES

[1] Soheil Abbasloo, Chen-Yu Yen, and H. Jonathan Chao. 2020. Classic Meets Modern: A Pragmatic Learning-Based Congestion Control for the Internet. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication* (Virtual Event, USA) *(SIGCOMM '20)*. Association for Computing Machinery, New York, NY, USA, 632–647. https://doi.org/10.1145/3387514.3405892

[2] Hao Chen, Xu Zhang, Yiling Xu, Ju Ren, Jingtao Fan, Zhan Ma, and Wenjun Zhang. 2019. T-Gaming: A Cost-Efficient Cloud Gaming System at Scale. *IEEE Transactions on Parallel and Distributed Systems* 30, 12 (2019), 2849–2865. https://doi.org/10.1109/TPDS.2019.2922205

[3] Cisco. 2022. *Global - 2022 Forecast Highlights*. Retrieved March 1, 2022 from https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_2022_Forecast_Highlights.pdf

[4] Christian Schroeder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviy-chuk, Philip H. S. Torr, Mingfei Sun, and Shimon Whiteson. 2020. Is Independent Learning All You Need in the StarCraft Multi-Agent Challenge? arXiv:2011.09533 [cs.AI]

[5] Sadjad Fouladi, John Emmons, Emre Orbay, Catherine Wu, Riad S. Wahby, and Keith Winstein. 2018. Salsify: Low-Latency Network Video through Tighter Integration between a Video Codec and a Transport Protocol. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. USENIX Association, Renton, WA, 267–282. https://www.usenix.org/conference/nsdi18/presentation/fouladi

[6] Gaetano, Luca De Cicco, Stefan Holmer, and Saverio Mascolo. 2017. Congestion Control for Web Real-Time Communication. *IEEE/ACM Trans. Netw.* 25, 5 (oct 2017), 2629–2642. https://doi.org/10.1109/TNET.2017.2703615

[7] Pablo Hernandez-Leal, Michael Kaisers, Tim Baarslag, and Enrique Munoz De Cote. 2017. A survey of learning in multiagent environments: Dealing with non-stationarity. *arXiv preprint arXiv:1707.09183* (2017).

[8] Tianchi Huang, Chao Zhou, Rui-Xiao Zhang, Chenglei Wu, Xin Yao, and Lifeng Sun. 2019. Comyco: Quality-Aware Adaptive Video Streaming via Imitation Learning. In *Proceedings of the 27th ACM International Conference on Multimedia* (Nice, France) *(MM '19)*. Association for Computing Machinery, New York, NY, USA, 429–437. https://doi.org/10.1145/3343031.3351014

[9] ITU-R BT.1788 2007. *Methodology for the subjective assessment of video quality in multimedia applications*. Standard. International Telecommunication Union, Geneva, CH.

[10] Bart Jansen, Timothy Goodwin, Varun Gupta, Fernando Kuipers, and Gil Zussman. 2018. Performance Evaluation of WebRTC-Based Video Conferencing. 45, 3 (mar 2018), 56–68. https://doi.org/10.1145/3199524.3199534

[11] Yueheng Li, Hao Chen, Bowei Xu, Zicheng Zhang, and Zhan Ma. 2023. Improving Adaptive Real-Time Video Communication Via Cross-layer Optimization. arXiv:2304.03505

[12] Ryan Lowe, YI WU, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/68a9750337a418a86fe06c1991a1d64c-Paper.pdf

[13] Zhan Ma, Meng Xu, Yen-Fu Ou, and Yao Wang. 2011. Modeling of rate and perceptual quality of compressed video as functions of frame rate and quantization stepsize and its applications. *IEEE Transactions on Circuits and Systems for Video Technology* 22, 5 (2011), 671–682.

[14] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural Adaptive Video Streaming with Pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (Los Angeles, CA, USA) *(SIGCOMM '17)*. Association for Computing Machinery, New York, NY, USA, 197–210. https://doi.org/10.1145/3098822.3098843

[15] Lifan Mei, Runchen Hu, Houwei Cao, Yong Liu, Zifan Han, Feng Li, and Jin Li. 2020. Realtime mobile bandwidth prediction using LSTM neural network and Bayesian fusion. *Computer Networks* 182 (2020), 107515. https://doi.org/10.1016/j.comnet.2020.107515

[16] Fangzhu Ming, Feng Gao, Kun Liu, and Chengmei Zhao. 2023. Cooperative modular reinforcement learning for large discrete action space problem. *Neural Networks* 161 (2023), 281–296. https://doi.org/10.1016/j.neunet.2023.01.046

[17] Arvind Narayanan, Xumiao Zhang, Ruiyang Zhu, Ahmad Hassan, Shuowei Jin, Xiao Zhu, Xiaoxuan Zhang, Denis Rybkin, Zhengxuan Yang, Zhuoqing Morley Mao, Feng Qian, and Zhi-Li Zhang. 2021. A Variegated Look at 5G in the Wild: Performance, Power, and QoE Implications. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference* (Virtual Event, USA) *(SIGCOMM '21)*. Association for Computing Machinery, New York, NY, USA, 610–625. https://doi.org/10.1145/3452296.3472923

[18] Yen-Fu Ou, Zhan Ma, Tao Liu, and Yao Wang. 2010. Perceptual quality assessment of video considering both frame rate and quantization artifacts. *IEEE Transactions on Circuits and Systems for Video Technology* 21, 3 (2010), 286–298.

[19] Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. 2021. Benchmarking Multi-Agent Deep Reinforcement Learning Algorithms in Cooperative Tasks. arXiv:2006.07869 [cs.LG]

[20] Reza Rassool. 2017. VMAF reproducibility: Validating a perceptual practical video quality metric. In *2017 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. 1–2. https://doi.org/10.1109/BMSB.2017.7986143

[21] Research and Markets. 2022. *Global Video Conferencing Market*. Retrieved March 1, 2022 from https://www.researchandmarkets.com/reports/5415505

[22] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2015. High-Dimensional Continuous Control Using Generalized Advantage Estimation. https://doi.org/10.48550/ARXIV.1506.02438

[23] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. https://doi.org/10.48550/ARXIV.1707.06347

[24] Jeroen van der Hooft, Stefano Petrangeli, Tim Wauters, Rafael Huysegems, Patrice Rondao Alface, Tom Bostoen, and Filip De Turck. 2016. HTTP/2-Based Adaptive Streaming of HEVC Video Over 4G/LTE Networks. *IEEE Communications Letters* 20, 11 (2016), 2177–2180. https://doi.org/10.1109/LCOMM.2016.2601087

[25] Weixun Wang, Tianpei Yang, Yong Liu, Jianye Hao, Xiaotian Hao, Yujing Hu, Yingfeng Chen, Changjie Fan, and Yang Gao. 2020. From Few to More: Large-Scale Dynamic Multiagent Curriculum Learning. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 05 (Apr. 2020), 7293–7300. https://doi.org/10.1609/aaai.v34i05.6221

[26] Yilin Wang, Sasi Inguva, and Balu Adsumilli. 2019. YouTube UGC Dataset for Video Compression Research. In *2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*. 1–5. https://doi.org/10.1109/MMSP.2019.8901772

[27] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing* 13, 4 (2004), 600–612. https://doi.org/10.1109/TIP.2003.819861

[28] WebRTC. 2021. *Real-time communication for the web*. Retrieved March 1, 2022 from https://webrtc.org/

[29] T. Wiegand, G.J. Sullivan, G. Bjontegaard, and A. Luthra. 2003. Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology* 13, 7 (2003), 560–576. https://doi.org/10.1109/TCSVT.2003.815165

[30] Jiachen Yang, Alireza Nakhaei, David Isele, Kikuo Fujimura, and Hongyuan Zha. 2018. Cm3: Cooperative multi-goal multi-stage multi-agent reinforcement learning. *arXiv preprint arXiv:1809.05188* (2018).

[31] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. 2015. A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication* (London, United Kingdom) *(SIGCOMM '15)*. Association for Computing Machinery, New York, NY, USA, 325–338. https://doi.org/10.1145/2785956.2787486

[32] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. 2021. The Surprising Effectiveness of PPO in Cooperative, Multi-Agent Games. https://doi.org/10.48550/ARXIV.2103.01955

[33] Huanhuan Zhang, Anfu Zhou, Yuhan Hu, Chaoyue Li, Guangping Wang, Xinyu Zhang, Huadong Ma, Leilei Wu, Aiyun Chen, and Changhui Wu. 2021. *Loki: Improving Long Tail Performance of Learning-Based Real-Time Video Adaptation by Fusing Rule-Based Models*. Association for Computing Machinery, New York, NY, USA, 775–788. https://doi.org/10.1145/3447993.3483259

[34] Anfu Zhou, Huanhuan Zhang, Guangyuan Su, Leilei Wu, Ruoxuan Ma, Zhen Meng, Xinyu Zhang, Xiufeng Xie, Huadong Ma, and Xiaojiang Chen. 2019. Learning to Coordinate Video Codec with Transport Protocol for Mobile Video Telephony. In *The 25th Annual International Conference on Mobile Computing and Networking* (Los Cabos, Mexico) *(MobiCom '19)*. Article 29, 16 pages. https://doi.org/10.1145/3300061.3345430