LiveStream Meta-DAMS: Multipath Scheduler using Hybrid Meta Reinforcement Learning for Live Video Streaming

Amir Sepahi, *Graduate Student Member, IEEE,* Lin Cai, *Fellow, IEEE,*, Wenjun Yang, *Graduate Student Member, IEEE,*, Jianping Pan, *Fellow, IEEE, Senior Member, ACM*

***Abstract*—Overcoming challenges in mobile environments, such as bandwidth constraints, user mobility, and network handoffs, is crucial for video streaming applications. To address these challenges, we can use multiple network paths to mitigate bandwidth limitations and guarantee end-to-end delay, enhancing the overall quality of experience for the users. This paper presents LiveStream Meta Learning-based Delay Aware Multipath Scheduler (LSMeta-DAMS), a novel learning-based multipath scheduler explicitly designed for live streaming applications. LSMeta-DAMS employs a hybrid meta-reinforcement learning architecture, incorporating both online and offline phases to enhance speed and accuracy for training and decision making. Prioritizing packet scheduling based on frame types and considering the video coding features like group of pictures (GOP), scalable video coding (SVC), and Dynamic Adaptive Streaming over HTTP (MPEG-DASH), LSMeta-DAMS offers a tailored solution for multipath video streaming. Trace-driven emulations highlight its superior performance, demonstrating up to 32% improvement in learning, up to 25% reduction in download time, up to 15% enhancement in video quality assessment, and up to 35% reduction in stalling time compared to the state-of-the-art multipath schedulers. These findings underscore LSMeta-DAMS's potential to substantially enhance video streaming experiences in highly dynamic network conditions.**

***Index Terms*—Meta reinforcement learning, multipath scheduler, live video streaming, delay guarantee**

I. Introduction

Video streaming has reshaped the landscape of mobile applications, providing users with unprecedented access to a variety of multimedia content anywhere, anytime. High bandwidth and low latency are the main requirements to have high-quality video streaming [1]. However, it is challenging to satisfy these requirements since wireless spectrum is scarce and expensive, and low latency and robust data transmission in mobile environment is difficult due to user mobility and channel impairments. The limitations of conventional protocols like TCP [2] in meeting the requirements of real-time applications, especially in terms of low latency and mobility support, have prompted the exploration of newer protocols. One such protocol is QUIC [3], a versatile transport-layer protocol supporting multiple streams and incorporating a 0-RTT/1-RTT handshake mechanism. Unlike TCP, QUIC can better support real-time applications through features like instant handshakes and accelerated loss detection and recovery. However, QUIC was originally designed for web applications, and it may not necessarily improve the performance of Dynamic Adaptive Streaming over HTTP (MPEG-DASH) video streaming [4].

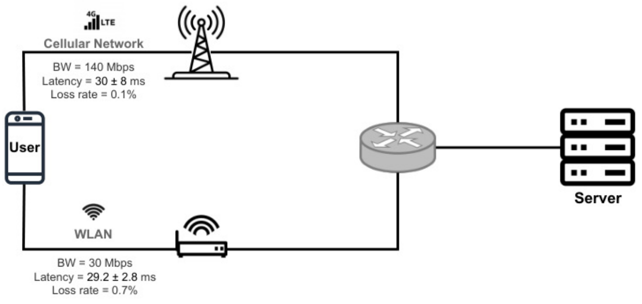


Fig. 1: An example of heterogeneous paths.

Given the increasing popularity of mobile services, many mobile devices are equipped with multiple wireless interfaces, such as 4G, 5G, and WiFi. As a result, multipath transport protocols such as multipath TCP (MPTCP) [5] and multipath QUIC (MPQUIC) [6] were developed to simultaneously use multiple access links and multiple network paths, resulting in faster and more reliable data transmission [7].

In addition, multipath transport protocols can provide seamless handover between network interfaces to better support user mobility [8]. They are resilient to link failures and can leverage bandwidth from multiple paths. However, using multipath protocols cannot always lead to higher quality services, particularly when the links (paths) are heterogeneous and highly dynamic (e.g., 5G and WLAN). An example of a heterogeneous scenario is shown in Fig. 1. Under these circumstances, multipath transport protocols face great challenges to provide low-latency transmission services for delay-sensitive applications, such as video streaming and online gaming.

One of the main reasons for this problem is out-of-order (OFO) packets. Specifically, if data packets arrive in a different order than intended, packets with higher sequence numbers must remain unprocessed in the receiver's buffer until lower sequence number packets arrive. This leads to wasted buffer space, head-of-line (HOL) blocking, unnecessary packet retransmissions, and lengthy reordering delays. In the context of video streaming, these issues collectively contribute to a degradation in video quality and playback experience.

The key challenge in mitigating out-of-order packets lies in devising an efficient scheduling policy to distribute data packets across available paths. This strategic distribution aims to enhance user perceived video quality, which depends on several performance metrics, including throughput, latency, and reliability. To achieve this objective, the development of a multipath (MP) packet scheduler is essential to effectively manage the distribution of packets across diverse network paths. However, designing an effective MP scheduler proves to be a challenging task, particularly when dealing with highly dynamic network conditions [9].

Existing MP schedulers can be categorized into model-based and learning-based types [1]. Model-based MP schedulers are rigid and cannot effectively handle highly dynamic network conditions. Learning-based schedulers offer agile responses, but they often rely on time-consuming Deep Q-Networks (DQN) [10] for online learning and may struggle with real-time adaptability. A main issue with the mentioned MP schedulers in both categories is that they are not specifically designed for delay-sensitive applications, such as live video streaming, where efficient scheduling of packets related to different video frames across available network paths is crucial for preventing degradation in video quality and ensuring a seamless playback experience. We discuss various types of schedulers in detail in Section II-E.

The design of an effective MP scheduler plays a crucial role in enhancing video streaming performance, particularly when utilizing multiple network paths (interfaces) simultaneously for transmission of multimedia data to the receiver. This becomes crucial when integrated with video coding and transmission technologies, such as group of pictures (GOP) [11], scalable video coding (SVC) [12], and MPEG-DASH [13]. Different from conventional MP schedulers, we present a learning-based MP scheduler explicitly designed to address the requirements of multipath video streaming. The main contributions of this paper are summarized as follows:

- We propose LiveStream Meta Learning-based Delay Aware Multipath Scheduler (LSMeta-DAMS), a learning-based MP scheduler explicitly crafted for live streaming applications, in conjunction with methods like GOP, SVC, and MPEG-DASH. This distinction sets LSMeta-DAMS apart from other MP schedulers.
- Considering GOP structure, we schedule packets related to different frame types on the path that ensures the earliest delivery to the client. This minimizes the packet reordering cost. We also implement a mechanism to prioritize I-frame packets over P- and B-frame packets. We incorporate a video chunk manager and a video frame manager into LSMeta-DAMS's framework to ensure it can determine the reception of each frame and each chunk.
- To make LSMeta-DAMS fast and accurate, we propose a hybrid meta-reinforcement learning (meta-RL) architecture, considering both online and offline meta-RL phases. Pretrained meta-models in the offline phase enhance both the learning process speed and the accuracy of the online learning phase. In the online learning phase of LSMeta-DAMS, we introduce a training algorithm built on top of the asynchronous advantage actor-critic (A3C) [14], which is faster and more accurate than Deep Q-Networks (DQN) widely used in other MP-schedulers.
- We implement LSMeta-DAMS using mpquic-go [15] for protocol aspects, TensorFlow [16], and keras-rl [17] for learning aspects. We utilize the free and open-source software project FFmpeg for video processing. Through trace-driven emulations, we demonstrate the superior performance of LSMeta-DAMS in terms of learning speed and accuracy, download time, and video quality assessment compared to state-of-the-art MP schedulers.

The rest of this paper is organized as follows. We first summarize the background and related work in Section II. We then specify the research problem and provide an overview of LSMeta-DAMS in Section III. We next detail the design of LSMeta-DAMS in Section IV. We present the experimental setup, LSMeta-DAMS configuration, and performance analysis in Section V. We finally conclude our work in Section VI.

II. Background and Related Work

In this section, we provide an overview of the background for LSMeta-DAMS and related work, including multipath transport protocols (Section II-A), video streaming technologies (Section II-B), stalling time (Section II-C), meta-reinforcement learning (Section II-D), and multipath packet scheduling (Section II-E).

A. Multipath Transport Protocols

Multipath transport protocols, initially stemming from the Stream Control Transmission Protocol (SCTP), used for signaling in telephone networks, evolved to address challenges in multipath communication [18]. Despite the difficulties encountered by MPTCP in gaining widespread adoption due to ossification issues, MPQUIC has emerged as a promising alternative. Developed as an extension of QUIC, MPQUIC demonstrates superior multipath performance over UDP and boasts flexibility in implementing various schedulers. Different from MPTCP, MPQUIC does not require modifications to operating systems or middleboxes, making it a more attractive option for modern networks. Both protocols enhance data transmission by leveraging multiple network paths, but MPQUIC's simplicity, resistance to middlebox tampering, and superior performance position it as a promising multipath transport protocol for diverse network environments.

In this paper, we take advantage of MPQUIC to design our delay-aware MP scheduler for delay-sensitive video applications, such as live video streaming and online gaming. Comparing MPQUIC and MPTCP (Fig. 2), both are designed to manage multipath communication, a feature that can substantially improve the performance and reliability of video streaming, particularly in mobile scenarios characterized by challenging network conditions. We use MPQUIC instead of MPTCP because it provides superior multipath performance over UDP, resulting in smoother video delivery, lower latency, and higher throughput. This is essential for a high-quality streaming experience. Moreover, MPQUIC's flexibility in scheduler implementation allows fine-tuning for optimal video content delivery, making it adaptable to the specific requirements of video streaming applications [19]. MPQUIC is also compatible with Dynamic Adaptive Streaming over

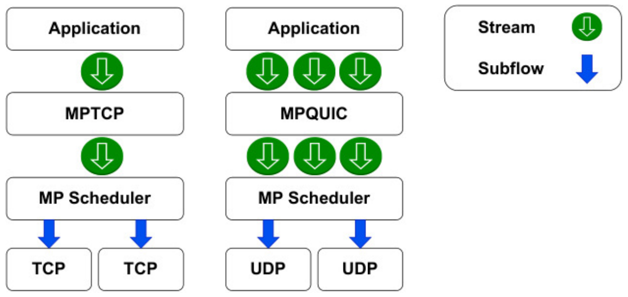


Fig. 2: MPTCP vs MPQUIC.

HTTP (DASH) which is the state-of-the-art standard of video streaming [13].

B. Video Streaming Technologies

Video content is a sequence of static images, or frames, displayed in rapid succession to create the illusion of motion. Several factors affect the size of a video file, including resolution, motion speed, frame rate, and the compression method used. In video compression and streaming, the GOP [11] method and SVC [12] have been widely deployed. GOP, also known as IPB, is a popular compression method where consecutive frames are grouped together and compressed as a unit. A GOP consists of an I frame that encodes the complete information for an image within the frame, followed by a variable number of P frames and B frames. P frames predict and record only the picture information that has changed from the previous I or P frame, while B frames use both the previous and subsequent frames as references.

SVC, an extension of the H.264 (MPEG-4 AVC) video compression standard, offers spatial, temporal, and quality scalability. This ensures graceful degradation in video quality when faced with channel loss [20]. The scalability allows for adaptability to varying network conditions, providing a more robust video streaming experience [21]. In this method, a video is encoded into multiple layers: a base layer that provides the lowest quality version of the video, and several enhancement layers that offer a higher quality version when decoded together with the base layer. This layered approach enables the video player to adapt to network conditions by selecting which layers to download and display based on network conditions. A general layered structure of SVC is shown in Fig. 3.

Together, these methods provide an effective solution for streaming video. SVC often works with adaptive streaming protocols like Dynamic Adaptive Streaming over HTTP (MPEG-DASH) which are designed to select layer(s) for video downloading based on network conditions [13]. MPEG-DASH breaks content up into a sequence of small HTTP-based segments, with each segment containing a short interval of playback time. Various bit rates (in layers) are available for the content. MPEG-DASH workflow is shown in Fig. 4.

MPEG-DASH starts with original video assets that are transcoded into multiple bitrates. Afterwards, these versions are packaged and encrypted using the MPEG-DASH standard, which segmented the video into small segments or chunks. These chunks, along with a Media Presentation Description

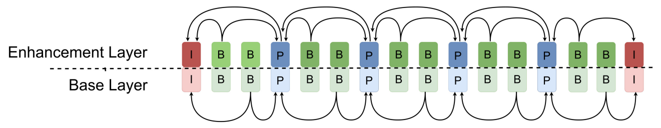


Fig. 3: A General Layered Structure of SVC.

(MPD) file, are stored on an origin server. In order to stream the video, the client device sends an HTTP request to a Content Delivery Network (CDN), which responds by sending the appropriate video chunks. In order to playback the video, the client device reassembles these chunks into a continuous video, automatically selecting the highest bitrate that can be downloaded in time for playback without causing buffering. This process allows MPEG-DASH to adapt to changing network conditions and provide a seamless quality of experience (QoE) for the users. MPEG-DASH should not be confused with a transport protocol - MPEG-DASH uses TCP or QUIC as its transport protocols. MPEG-DASH uses existing HTTP web server infrastructure that is used for delivery of essentially all World Wide Web content.

C. Stalling Time

A key performance metric for video streaming applications is stalling time, which results from pauses or buffering interruptions during playback, frustrating users with disruptions. This occurs when the video player's buffer lacks sufficient data to play continuously, often influenced by factors such as network congestion, server issues, client-side problems, and Adaptive Bitrate Streaming (ABR) adjustments [22]. Fluctuations in network conditions, inadequate server performance, device or player glitches, and ABR algorithms dynamically adjusting video quality based on varying network conditions contribute to the occurrence of stalling. Mitigating this issue involves optimizing streaming protocols, implementing effective buffering strategies, and enhancing network infrastructure to ensure a seamless and uninterrupted video playback experience [23].

Utilizing MP transport protocols like MPQUIC and implementing an effective MP scheduler in video streaming system design can significantly diminish stalling time. A well-designed MP scheduler optimally allocates data across available paths, ensuring load balancing and adaptiveness to changing network conditions, thereby preventing over-reliance on a single path prone to stalling. This approach not only minimizes the risk of buffer depletion but also proactively fetches and buffers content, improving the overall robustness of the streaming system and elevating the quality of the user experience.

D. Meta-Reinforcement Learning

Deep Reinforcement Learning (DRL) approaches like DQN [10] and actor-critic methods [14] can perform well in many challenging and dynamic environments after extensive training. Human intelligence, on the other hand, can understand and adapt to new tasks quickly by leveraging their prior experiences with similar tasks. This inspires meta-reinforcement learning (meta-RL) [24].

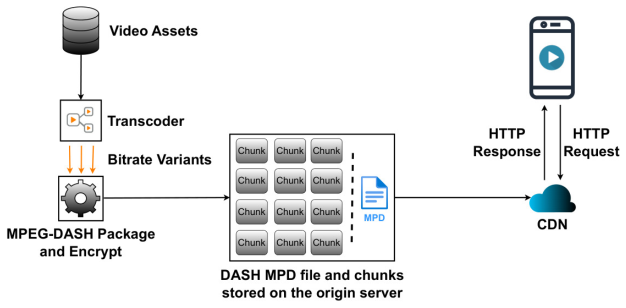

Video Assets
Transcoder
Bitrate Variants
MPEG-DASH Package and Encrypt
Chunk
MPD
DASH MPD file and chunks stored on the origin server
HTTP Response
HTTP Request
CDN

Fig. 4: MPEG-DASH Workflow.

In the realm of artificial intelligence (AI), the integration of the meta-learning concept has been successful [25]. It allows deep learning models to enhance their learning algorithms iteratively over a number of learning episodes. This concept is also applicable to RL, a process akin to how we engage in multiple learning tasks concurrently in our daily lives. In meta-RL, different RL algorithms operate at different timescales but in parallel. Using the meta-RL framework, Deep RL models engage in multiple RL tasks simultaneously, each unfolding at distinct timescales.

Motivated by the capability of meta-RL, we integrate it into the architecture of our proposed MP scheduler. Traditional RL algorithms often experience difficulty in learning and adapting to new tasks efficiently. As a result, they typically require extensive training from scratch, which is both time-consuming and computationally expensive. The inability to adapt quickly to new tasks can be a significant disadvantage in highly dynamic environments.

Meta-RL can train models to be adaptive to new tasks rapidly without the need to retrain them from scratch, leveraging the knowledge gained from previous tasks. This accelerates the learning process and allows the scheduler to be more responsive to highly dynamic network conditions [1].

E. MP Packet Schedulers

The MP scheduler is responsible for distributing packets over the available subflows. In the following section, we categorize MP schedulers into two groups: model-based and learning-based MP schedulers.

Model-based schedulers: A model-based multipath packet scheduler, such as minRTT, Round Robin (RR), Blocking Estimation (BLEST) [26], and Earliest Completion First (ECF) [27], uses a structured and rule-driven approach for packet distribution across available subflows. These schedulers are based on predefined rulesets and models that enable them to make informed decisions based on well-defined criteria. For instance, minRTT prioritizes subflows with minimum RTTs, while RR cyclically distributes packets among available paths.

BLEST and ECF implement sophisticated algorithms based on bandwidth estimation and the available congestion window. The deterministic nature of model-based approaches facilitates predictability in network management, offering stability in scenarios with relatively stable and well-defined network conditions. However, their inherent rigidity becomes apparent in highly dynamic or unpredictable environments, where adaptability may be compromised.

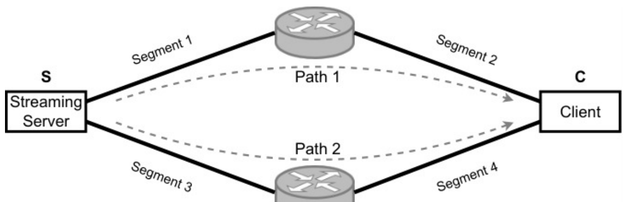


Fig. 5: Multipath Video Streaming.

Learning-based schedulers: Learning-based schedulers, such as Peekaboo [28], M-Peekaboo [9], RL [29], and Reles [30], can adapt their policies to new network conditions. This makes them more suitable for highly dynamic environments. While providing flexibility and adaptability, these schedulers often face challenges, such as the time-consuming training of DQN, which may impede real-time adaptability.

Learning-based algorithms come in two varieties: online learning and batch learning (offline learning). For batch learning (offline learning), any improvements to the model requires retraining over the entire dataset, making it less adaptive and requiring sufficient storage capacity to store the entire dataset. On the other hand, online learning can adapt to data with varying patterns and distributions as they emerge, making it more suitable for real-time tasks such as multipath scheduling. However, a trade-off exists between accuracy and convergence time [31]. Online learning algorithms with less complex architectures are faster but less accurate, and vice versa.

To address the challenges related to DQN convergence speed and the limitations of both online and offline learning, meta-DAMS [1] was developed, leveraging meta-RL. A hybrid meta-RL architecture was proposed, considering both offline and online meta-RL. A3C was used as the learning algorithm for the MP scheduler. A3C adopts an actor-critic framework, incorporating asynchronous training and an advantage function to enhance the algorithm's training efficiency across parallel threads. This results in faster and more accurate learning compared to DQN.

III. Problem Description and Solution Framework

In this section, we first explain the multipath video streaming system model. We then describe the challenges of multipath video streaming. Afterwards, we describe how we intend to resolve those problems. For easy presentation, we consider only the two-path scenario presented in Fig. 5 in the problem formulation. It is straightforward to extend the method to scenarios with a larger number of paths.

A. Multipath Video Streaming

In multipath video streaming, multiple paths are used to reduce the impact of network congestion or disruptions on the viewing experience. This approach increases robustness and ensures that video playback is smoother even in challenging network conditions [32].

Considering the multipath network topology depicted in Fig. 5, the client C initiates a media stream request from the streaming server S. The server delivers the requested bitstream through two separate paths, each comprising two segments connected through intermediate nodes. A buffering delay may be introduced by intermediate nodes, which forward packets from the server to the client. Buffer interfaces deploy first-in, first-out (FIFO) queues. As links/paths have variable data rates and congestion levels, the sending rate from the server changes over time. On the client side, an initial playback delay δ is experienced following the stream request. The client then begins to decode the media stream and plays the video continuously. Wireless video transmission through multiple interfaces is our main application. During the streaming session, server selects to send a subset of pre-encoded media packets for communication with the client, depending on the available bandwidth on each path.

B. Streaming Model

The video sequence is encoded into a bitstream using a scalable (layered) video encoder, consisting of a base layer and several enhancement layers. The bitstream is then fragmented into network packets in accordance with two main rules. First, each network packet contains data relevant to at most one video frame. This is due to the fact that we use GOP as our compression method in this paper. According to Section II-B, in this method, there are three types of frames, I, P, and B. A fragmented packet can be related to at most one of these types. Second, an encoded video frame can be divided into multiple network packets. Here, the video data is classified into a base layer, containing essential information for reconstruction, and enhancement layers, which provide additional details for higher quality. Consider a system where the client has $\mathcal{F} = \{1, \ldots, f, \ldots, F\}$ flow requests. Suppose there is a streaming request S associated with one of these flows ($f \in \mathcal{F}$). S contains a number of packets, denoted by $\mathcal{N} = \{1, \ldots, n, \ldots, N\}$, which need to be transmitted over $\mathcal{P}$ interfaces (paths) denoted by $\mathcal{P} = \{1, ..., p, ..., P\}$. Each network packet, n, is characterized by its size s_n in bytes, and its decoding timestamp t_n^d.

In the GOP structure, I frames hold greater significance for the client compared to P and B frames. If the client encounters difficulty decoding an I frame, the subsequent P and B frames in the corresponding GOP lose their utility. In our streaming solution, I-frame packets have the highest importance and, therefore, have the highest utility. We define the utility of each network packet, λ_n, which represents the reduction in distortion perceived by the client when the packet is successfully decoded [33]. The value of λ_n is determined by the importance of the packet's content to the overall quality of the video stream. For example, packets containing I frames, which are critical for reconstructing video frames, have higher values of λ_n, reflecting their greater impact on video quality. In contrast, packets containing P or B frames have lower values of λ_n since they depend on I frames and contribute less to the immediate reconstruction quality.

The streaming server uses an MPQUIC scheduler to determine which packets ($\mathcal{N}^* \subset \mathcal{N}$) should be scheduled on path $p \in \mathcal{P}$ when the path has an available window ($\omega_p(t) > 0$) [1]. The MPQUIC scheduler updates its scheduling policy $\Pi = \{\pi_1, \ldots, \pi_n, \ldots, \pi_N\}$ according to the current network condition. Each policy π_n corresponds to the transmission decision for the n-th packet, consisting of $[a_n, t_n^s]$, where a_n is the action chosen for the n-th packet and t_n^s is its sending time. The scheduling policy keeps updating until the transmission of all packets ($|\mathcal{N}|$) is completed.

The arriving time of the n-th packet through path p can be defined as:

$$t_n^a = t_n^s + OWD_p + d_p^t + \eta_p, \tag{1}$$

where OWD_p is the one-way delay of path p, assumed to be half of τ_p, d_p^t is the transmission delay at the sender, and η_p is the total retransmission delay if the packet is lost during the first attempt. In our streaming model, the receiver can successfully decode the n-th packet only if its arrival time (t_n^a) is smaller than its decoding deadline (δ), i.e., if $t_n^a \leq t_n^d + \delta$. Since I-frame packets are more important to QoE, they have a more stringent decoding deadline.

The optimization framework inherently integrates key constraints essential for ensuring high-quality multipath video streaming. For this purpose, we introduce a binary variable, $\zeta(\pi_n)$, which indicates whether the n-th packet reaches the decoder prior to its decoding deadline and if all its preceding dependent packets are successfully decoded according to the streaming policy $\pi_n \in \Pi$. If $\zeta(\pi_n)$ is equal to 1, it signifies that the packet arrives at the decoder on time, and all its preceding dependencies have been successfully decoded. Furthermore, we consider the distinction between the order of frames in the bitstream and the order in which they are decoded at the client, impacting the scheduling of a B frame positioned in the bitstream after the upcoming P frame it relies on. This parameter effectively embeds constraints related to decoding deadlines, network conditions (such as available bandwidth and latency), and interdependencies between frame types, thereby guiding the optimization process toward maximizing video quality while maintaining adaptability in dynamic network environments.

Given the utility value λ_n for each packet, the total benefit of a streaming policy $\pi_n \in \Pi$, denoted by $\Psi(\pi_n)$, represents the overall quality experienced by the receiver and can be calculated as the sum of utilities of all successfully decoded packets:

$$\Psi(\pi_n) = \sum_{\forall n : \zeta(\pi_n) = 1} \lambda_n. \tag{2}$$

This formulation ensures that the optimization objective of maximizing the client's perceived video quality is achieved by prioritizing packets that contribute the most to reducing distortion.

C. Problem Statement

A multipath scheduler takes packets from applications and determines on which subflow to transmit each packet. In multipath video streaming, packets can vary significantly in importance and urgency due to their types, which include base layer (BL) and enhancement layer (EL) packets of different frame types (I, P, and B frames). The scheduler's primary goal is to find the optimal streaming policy π^* to maximize the total utility (Ψ) experienced by the client, which is expressed as:

$$\Psi\left(\pi^*\right) = \max_{\pi \in \Pi} \sum_{\forall n : \zeta(\pi_n)=1} \lambda_n . \tag{3}$$

This optimization problem is NP-complete [34]. Our solution introduces an MP scheduler that leverages hybrid meta-reinforcement learning to update the scheduling policy (Π) efficiently and accurately in response to dynamically changing network conditions. In this approach, we consider the decoding deadlines for various types of packets, addressing the specific requirements of video streaming.

The main challenges addressed by this problem include adapting to highly dynamic network conditions, where bandwidth, latency, and packet loss vary unpredictably, and prioritizing packets of different importance and urgency (e.g., I, P, and B frames) to ensure timely delivery and minimize distortion. Additionally, the scheduler must handle the computational complexity of real-time decision-making while operating with minimal prior knowledge of network behavior, making it robust across various environments. To address these challenges, our proposed scheduler uses hybrid meta-reinforcement learning, which combines offline meta-learning with online reinforcement learning (RL) to quickly adapt to network changes, balance multiple objectives, and achieve near-optimal performance efficiently.

D. Solution Overview

LSMeta-DAMS employs meta-RL and DRL techniques to efficiently address the rapid adaptation to new policies with a limited number of training episodes. By doing so, our approach facilitates accelerated learning with minimum amount of training data.

Fig. 6 depicts the flow chart of our solution. The client initiates an HTTP request to access the desired video content. On the server side, IPB frames from the requested video content are extracted and packetized in the application layer. Throughout this process, packets are labeled to denote the respective frame type (I, P, or B frame) and the associated layer (BL or EL). This labeling ensures precise identification and decoding on the client side, thereby facilitating the reconstruction of the video content with due consideration to the layered structure. Moving to the transport layer, LSMeta-DAMS schedules packets, associated with different frames and layers, across available subflows according to the current network condition. The generation of scheduling policies is executed by a neural network embedded in our scheduler, which takes input from observations derived from the environment, meta data, and meta models. LSMeta-DAMS enhances MPQUIC in several ways. First, it understands the preference for multiple paths and leverages this knowledge to optimize overall performance. Second, it is aware of the deadline for each video chunk. Third, it takes advantage of hybrid meta-RL in its architecture, using both online and offline meta-RL. Finally, it utilizes meta data obtained from pre-trained meta models and the client side. Based on this information,

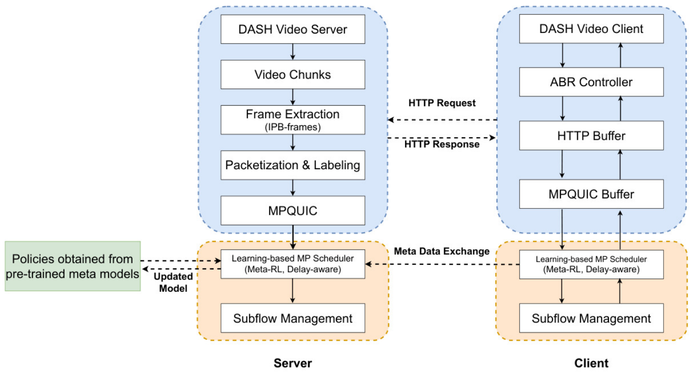


Fig. 6: General Workflow.

the scheduler manages network paths quickly and accurately in response to highly dynamic network conditions to meet deadlines and satisfy user preferences, as detailed in Section IV.

IV. LSMeta-DAMS Design

In this section, we first explain meta-RL in Section IV-A. Then, we discuss hybrid meta-RL to clarify why it is a suitable approach to consider in our algorithm design in Section IV-B. Next, we describe the various learning elements that are required to configure LSMeta-DAMS operations in Section IV-C.

A. Meta-Reinforcement Learning

Our proposed MP scheduler leverages the architecture of meta-RL to train models that can adapt to new tasks rapidly and accurately. Unlike classical RL, meta-RL offers a distribution of Markov decision processes (MDP). Each MDP is referred to as a task [35], and we consider a distribution over these tasks $p_{\mathcal{T}}(\cdot)$. Each task $\mathcal{T} = (\mathcal{S}, \mathcal{A}, r, \gamma, p_0, p_d)$ includes state space $\mathcal{S}$, action space $\mathcal{A}$, reward function r, discount factor γ, initial state distribution $p_0\left(\mathbf{s}_0,\right)$, and transition distributions $p_d\left(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t\right)$.

There are two steps in meta-RL: meta-training, where the agent learns an algorithm, and meta-testing, where the agent applies this algorithm to determine an optimal policy π_i^* for the current task $\mathcal{T}_i$. However, the main goal is to find an optimal policy π^* for a distribution of tasks ($p_{\mathcal{T}}(\cdot)$) to maximize the expected return $J(\theta)$ over tasks:

$$\max_{\theta} J(\theta) = \mathbb{E}_{\mathcal{T} \sim p}\left[J_{\mathcal{T}}(\theta)\right], \tag{4}$$

where $J_{\mathcal{T}}(\theta)$ is the expected discounted return:

$$J_{\mathcal{T}}(\theta) = \mathbb{E}_{s_0 \sim d_0, a_t \sim \pi_\theta(s_t), s_{t+1} \sim p_d(s_t, a_t)}\left[\sum_t \gamma^t r_t\right]. \tag{5}$$

In practice, the meta dataset $\mathcal{D}$ consists of a set of tasks divided into two subsets: meta-training tasks $\mathcal{D}^{(train)}$ and meta-testing tasks $\mathcal{D}^{(test)}$. Agent uses a set of meta-training tasks $\mathcal{D}^{(train)}$ to learn a stochastic algorithm Γ while interacting with the environment to learn a policy π_i to maximize the objective function for each task $J_{\mathcal{T}_i}$:

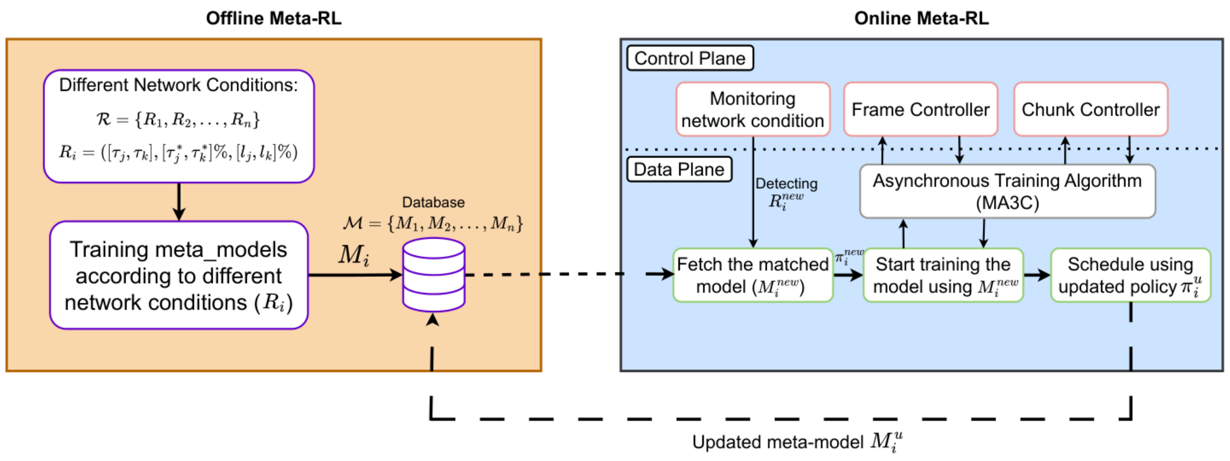

Offline Meta-RL
Different Network Conditions:
$\mathcal{R} = \{R_1, R_2, \ldots, R_n\}$
$R_i = ([\tau_j, \tau_k], [\tau_j^*, \tau_k^*]\%, [l_j, l_k]\%)$
Training meta_models according to different network conditions (R_i)
M_i
Database
$\mathcal{M} = \{M_1, M_2, \ldots, M_n\}$
Online Meta-RL
Control Plane
Monitoring network condition
Frame Controller
Chunk Controller
Data Plane
Detecting R_i^{new}
Asynchronous Training Algorithm (MA3C)
Fetch the matched model (M_i^{new})
π_i^{new}
Start training the model using M_i^{new}
Schedule using updated policy π_i^u
Updated meta-model M_i^u

Fig. 7: LSMeta-DAMS: High-Level Architecture.

$$\mathbb{E}_{\mathcal{T}_i \sim \mathcal{D}^{(\text{train})}} \left[\mathbb{E}_{\tilde{\pi}_i \sim -(\mathcal{T}_i)} \mathcal{J}_{\mathcal{T}_i} (\pi_i) \right] . \tag{6}$$

During testing, the agent (scheduler) samples a task $\mathcal{T}_i$ with a set of meta-testing tasks $\mathcal{D}^{(test)}$ to learn π_i using Γ and to apply the learned policy π_i in the environment. Meta-training and meta-testing are carried out in both offline and online meta-RL.

B. Hybrid Meta-RL

To enhance the accuracy and speed of our scheduler, we propose a hybrid meta-reinforcement learning (meta-RL) approach for designing LSMeta-DAMS. Fig. 7 illustrates the LSMeta-DAMS architecture, comprising both offline and online meta-RL phases. Offline meta-RL refers to the phase where the meta-models are pre-trained using historical data and simulated environments before deployment. This phase aims to create a set of generalized models, called meta-models, that are capable of providing a good starting point for learning under various network conditions. In contrast, online meta-RL involves using these pre-trained meta-models during actual deployment to quickly adapt to real-time changes in the network environment. This phase focuses on fine-tuning and optimizing the scheduling policy in response to the current network conditions, ensuring efficient and accurate decision-making in a dynamic environment. The online and offline phases exchange their updated experiences with each other as follows:

1) Offline meta-RL: The main purpose of this phase is to train a set of meta-models $\mathcal{M} = \{M_1, M_2, \ldots, M_n\}$. Each M_i represents a meta-model derived for a specific network condition that LSMeta-DAMS may encounter in online phase. The mean RTT, the jitter (RTT variation rate), and the packet loss rate are indicators of network conditions. We consider different ranges for RTT, jitter and packet loss to distinguish different network conditions. $\mathcal{R} = \{R_1, R_2, \ldots, R_n\}$ represents a set of different ranges for different network conditions. Each meta-model M_i is derived based on a specific range of these parameters. For instance, meta-model M_i is derived considering RTT between $[\tau_i, \tau_{i+1}]$, jitter between $[\tau_i^*, \tau_{i+1}^*]\%$, and packet loss rate between $[l_i, l_{i+1}]\%$ [1]. To pre-determine these network conditions, we use statistical analysis based on historical data and real-world network measurements to define ranges for RTT, jitter, and packet loss that cover most typical scenarios encountered in practical settings. This method allows us to establish a set of representative meta-models that reflect a wide spectrum of possible network conditions. However, network environments are inherently dynamic, and conditions may occasionally fall outside these predefined ranges. In such cases, LSMeta-DAMS employs an interpolation strategy to select the most appropriate meta-model. When the observed network condition does not match any predefined range exactly, the system calculates the weighted average of the scheduling policies from the two closest meta-models based on their similarity to the current network state. This adaptive approach ensures that LSMeta-DAMS can handle dynamic network conditions effectively. With the help of this classification, we are able to cover most network conditions that may occur in a dynamic environment. Therefore, considering each network condition as a task $\mathcal{T}_i$, meta-training and meta-testing are applied to derive each meta-model as well as the optimal scheduling policy π_i^* for each task in offline meta-RL. Then in the online phase, when LSMeta-DAMS encounters a specific network condition, it fetches the associated meta-model corresponding to that condition, and attempts to converge to the optimal scheduling policy using that meta-model. As a result, LSMeta-DAMS is able to converge faster without compromising its accuracy.

2) Online meta-RL: As depicted in Fig. 7, LSMeta-DAMS operates with both a control plane and a data plane during the online phase. These planes consist of several blocks working collaboratively to retrieve the matched meta-model corresponding to the new network condition from offline phase database. Once retrieved, this meta-model is used to quickly and accurately update the online model to adjust the scheduling policy effectively in each time slot. We explain the online meta-RL phase function in details in the following.

There is a block in the control plane that constantly monitors the network conditions to detect any changes. If the new network condition falls outside the current range (R_i), it sends a control signal to the fetch block in the data plane. This signal instructs the fetch block to retrieve the corresponding meta-model (M_i^{new}) from the offline phase database. After meta-model selection, LSMeta-DAMS is able to fine-tune the selected meta-model (M_i^{new}) using few-shot learning [36] and derive the scheduling policy using MA3C. MA3C is the learning algorithm that we developed on top of A3C, detailed in Section IV-C.

Since our application is video streaming, to make online model training more accurate, we assigned two other controllers in the control plane to inform the MA3C about the reception of each video frame and video chunk, helping the learning algorithm in deriving a more precise scheduling policy. As a final step, LSMeta-DAMS deploys and uses the new policy until a new change is detected. As soon as a new network condition is detected, the updated meta-model M_i^{u} is sent back to the $\mathcal{M}$ in the offline meta-RL phase.

C. Reinforcement Learning Indicators

As discussed earlier in Section IV-B, LSMeta-DAMS utilizes a modified version of A3C algorithm (MA3C) to generate the policy during operation. Additionally, it employs a hybrid meta-RL approach to converge rapidly to the optimal policy without compromising accuracy. Consequently, the entire framework can be regarded as an MDP, which LSMeta-DAMS can solve through the coupling of meta-RL using MA3C. Detailed information about the framework's learning indicators is provided below.

State: In an MDP, a system state refers to the information captured in a snapshot of the environment that an agent can perceive during the learning process. LSMeta-DAMS represents the learning agent in our scenario. General state is determined by the parameters of the transport layer of the available paths ($\mathcal{P}$) including number of inflight packets (Inf_p), throughput measurement (T_p), congestion window (w_p), size of the send window ($swnd_p$), RTT (τ_p) and packet loss (l_p). The agent observes the current state of the system in each episode or scheduling interval (SI). Assuming that in the t-th SI, the system state is denoted by $s_t = (s_{t,1}, s_{t,2}, \dots, s_{t,n})$, where $s_{t,i}$ $(1 \leq i \leq n)$ signifies the observed state of the i-th subflow, represented as a tuple $s_{t,i} = (T_{t,i}, swnd_{t,i}, Inf_{t,i}, w_{t,i}, \tau_{t,i}, l_{t,i})$, where

- $T_{t,i}$ is the subflow throughput measurement;
- $swnd_{t,i}$ is the maximum amount of unacknowledged data that a sender can have in-flight;
- $Inf_{t,i}$ is the number of unacked packets;
- $w_{t,i}$ is the average congestion window sizes;
- $\tau_{t,i}$ is the subflow mean RTT;
- $l_{t,i}$ is the packet loss of the subflow.

Action: The action space in our scenario is discrete and its size is determined by the available paths ($\mathcal{A} = \{a_0, \dots, a_i, \dots, a_P\}$). Each time LSMeta-DAMS deploys a scheduling policy, it can choose from this set of actions, and for each action it receives a reward. Each scheduling policy is executed for a specific amount of time (T), where $T = \min\{\tau_0, \dots, \tau_i, \dots, \tau_P\}$. However, if the sender receives an acknowledgment packet (ACK) before T, it proceeds to choose the next action.

As an example, in the case where a user has only two paths to transmit its data, our action space would be $\mathcal{A} = \{a_0, a_1, a_2\}$ where a_0 corresponds to the option of waiting for at least one path to become available. The action a_1 involves deciding whether to transmit data through the currently available path or to wait for the other path to become active. Finally, the action a_2 entails the selection of packets for transmission from both available paths.

The data collection occurs in two instances [29]. First, during the scheduling process, a data tuple [State (T ms before), Action (T ms before), Reward (T ms during), State (current)] is gathered every T ms. This occurs if no acknowledgment (ACK) is received before T ms, under the condition that the chunk download is not yet completed. Second, upon the completion of the chunk download (or episode), a data tuple [State (T ms before), Action (T ms before), Reward (episode), and State (current)] is collected.

Reward: At any given time t, the reward $r_t \in \mathbb{R}$ represents the performance of the agent. This scalar reward signal comprises both positive and negative values which correspond to rewards and penalties, respectively. Our reward $\mathbb{R}$ function comprises frame-level (R_f) and chunk-level (R_c) components ($\mathbb{R} = f(R_f, R_c)$). To calculate R_f and R_c, we implement a frame controller and chunk controller in LSMeta-DAMS architecture. The details of the frame controller and chunk controller are discussed in Section IV-B.

The frame-level reward (R_f) focuses on the timely delivery of I-frame packets, which are critical for video quality, and the successful delivery of P-frame and B-frame packets. The reward for I-frames is given by:

$$R_f = \sum_{i \in \text{I-frames}} \Big(R_f^I \cdot \mathbb{I}(t_i^a \leq t_i^d + \delta) - P_f^I \cdot \mathbb{I}(t_i^a > t_i^d + \delta) \Big) + \sum_{j \in \text{P,B-frames}} G_j, \quad (7)$$

where R_f^I is the positive reward for timely delivery of I-frames, P_f^I is the penalty for delayed or lost I-frames, G_j is the goodput contribution of P-frame and B-frame packets, and $\mathbb{I}(\cdot)$ is an indicator function that returns 1 if the condition is met and 0 otherwise.

The chunk-level reward (R_c) ensures efficient download of video chunks by providing a higher reward for faster downloads. It is defined as:

$$R_c = \frac{C_{\text{max}}}{C_{\text{download}}},$$

where C_{max} is a constant representing the maximum reward for the fastest possible download, and C_{download} is the actual time taken to download the chunk.

To account for both immediate and future rewards, we employ a discounted future reward approach. This method allows the agent to consider both immediate and long-term benefits, which is crucial for learning an effective policy over time. The discounted future reward is calculated as:

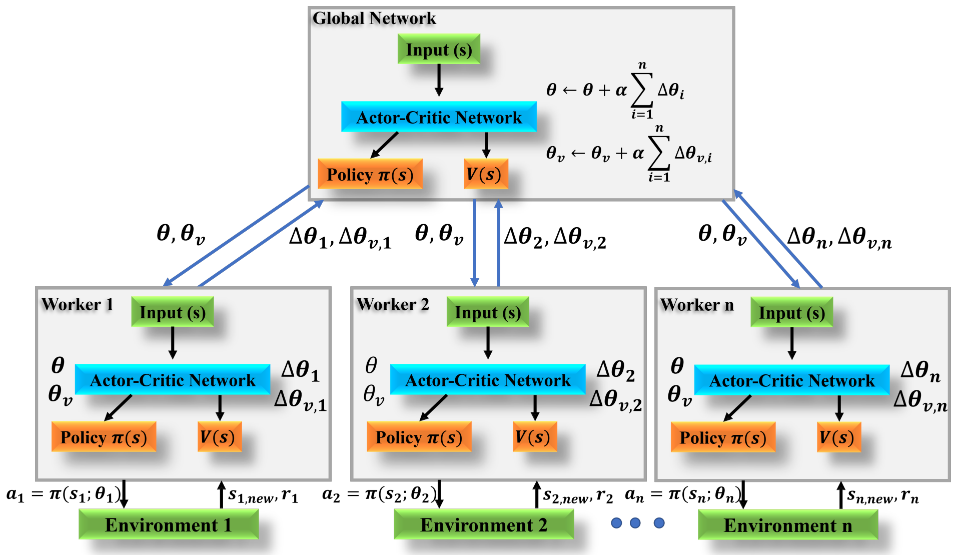


Fig. 8: High-Level Architecture of the MA3C Algorithm in LSMeta-DAMS. The global network sends the current policy and value network parameters (θ and θ_v) to multiple worker agents (1, 2, ..., n). Each worker interacts with its environment, taking actions ($a_i = \pi(s_i; \theta_i)$) based on the current policy, and receiving new states and rewards ($s_{i,\text{new}}, r_i$) from the environment. The workers compute gradients ($\Delta\theta_i, \Delta\theta_{v,i}$) using their experiences and send them back to the global network, which aggregates these gradients to update the global parameters. This process allows for efficient parallel training and faster convergence in dynamic network conditions.

$$R_t\left(s_t, a_t\right) = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \tag{8}$$

where $\gamma \in (0,1)$ is the discount factor. This structure ensures that the agent's decisions align with the long-term objectives of high-quality video streaming by prioritizing timely delivery, minimizing delays, and optimizing the overall QoE.

Asynchronous Training Algorithm: LSMeta-DAMS employs a modified asynchronous training algorithm known as modified-A3C (MA3C), which we developed as an extension of the A3C algorithm. In contrast to previous approaches that commonly utilize DQN as their online learning module, our work introduces enhancements to the basic A3C framework to better align it with the requirements of our application, particularly in the context of multipath packet scheduling. A high-level architecture of A3C is shown in Fig. 8. In the following, we first provide an overview of the A3C algorithm. Following that, we discuss the modifications implemented in A3C to introduce MA3C.

1) A3C: A3C has several advantages over DQN. A3C adopts the actor-critic framework and incorporates asynchronous training and an advantage function to enhance the algorithm's training efficiency across parallel threads. In this setup, the actor network generates actions or policies, while the critic network assesses the chosen actions. This design accelerates the overall training speed of the algorithm. A3C operates without the need for experience replay, and its asynchronous training ensures a diverse exploration and exploitation strategy. By allowing each agent thread to interact with the environment in parallel, the correlation among training samples is reduced, enhancing the agent's learning speed. Moreover, A3C can be trained two times faster than DQN, even if a multi-core CPU is used rather than a GPU [14].

A3C maintains a policy $\pi\left(a_t \mid s_t; \theta\right)$ and an estimate of the value function $V\left(s_t; \theta_v\right)$. Updates to the policy and value function occur either after a fixed number of steps or when a terminal state is reached. In cases where the action function $Q(a_t, s_t)$ is non-negative, the policy gradient becomes greater than or equal to zero, leading to increased action probabilities and enlarged policy variance, thereby slowing down the agent's learning. To overcome this challenge, A3C introduces an advantage function given by

$$A(a_t, s_t; \theta) = \sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k V\left(s_{t+k}; \theta_v\right) - V\left(s_t; \theta_v\right), \tag{9}$$

where k varies from state to state and is capped by $t_{\max}$. If $A(a_t, s_t; \theta) > 0$, the probability of action a_t increases; otherwise, it decreases.

Recognizing that entropy can gauge the uncertainty of a probability distribution, A3C incorporates policy entropy into its framework. A higher entropy serves to prevent A3C from converging to a suboptimal policy. Consequently, the policy gradient update for A3C is defined as follows:

$$\begin{aligned} d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi\left(a_i \mid s_i; \theta'\right)\left(R - V\left(s_i; \theta'_v\right)\right) \\ + \beta \nabla_{\theta'} H\left(\pi\left(s_t; \theta'\right)\right), \\ d\theta_v \leftarrow d\theta_v + \partial\left(R - V\left(s_i; \theta'_v\right)\right) / \partial\theta'_v, \end{aligned} \tag{10}$$

where θ and θ_v are the global parameters regarding to the action estimation and value estimation, respectively. θ' and θ'_v

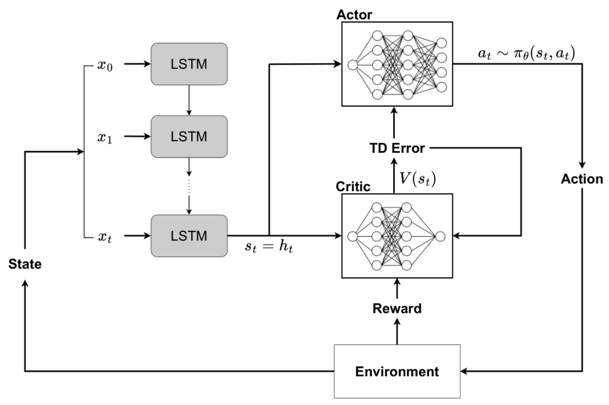


Fig. 9: LSTM integration in MA3C algorithm.

are the local parameters regarding to the action estimation and value estimation, respectively.

2) MA3C: On top of the original A3C, the proposed MA3C changes the optimization method used in A3C and incorporates a Long Short-Term Memory (LSTM) [37] model into the design.

While RMSProp [38] is employed in the original A3C design, MA3C utilizes the Adam optimizer [39] instead of RMSProp. Adam combines ideas from both momentum and RMSProp, utilizing moment estimates and an adaptive learning rate to optimize the loss function. As the effective steps in the parameter space of the reinforcement learning model, performed with the gradient moment estimates, are bounded by the learning rate (α), the Adam method demonstrates resilience to sudden changes in the magnitude of the gradients of the loss function. This characteristic reduces the sensitivity to abrupt changes, making Adam less dependent on extensive hyperparameter searches for viable learning rates [39].

In problems where temporal dependencies exist between a received reward at timestep t and information from previous inputs, there is an incentive to maintain knowledge of input history. One approach to sustain the context of a problem over a sequence of time is by using recurrent neural networks (RNNs). In the context of deep learning, a widely-used RNN architecture is the LSTM model, which belongs to a set of RNN models referred to as gated models.

Utilizing LSTM in the design of MA3C for multipath packet scheduling is justified based on the temporal nature of the input state, encompassing components such as throughput, unacknowledged data, congestion window size, mean RTT, and packet loss. LSTM excels in capturing temporal dependencies, facilitating sequential decision-making by the MA3C agent. Its ability to handle variable-length sequences and preserve memory and context makes it well-suited for scenarios where decisions depend on historical context. Additionally, LSTM enhances the model's capacity to learn patterns in sequential data, contributing to the effective adaptation of scheduling policies in dynamic multipath environments.

Fig. 9 illustrates the integration of LSTM within the MA3C architecture, demonstrating how LSTM is embedded in both the worker agents and the global network. This integration allows each block of the MA3C framework to effectively leverage temporal dependencies, thereby improving learning and scheduling decisions by maintaining context across sequential data inputs. The figure shows how LSTM units are used to process sequences of input states, capture historical information, and enhance the overall performance of the MA3C algorithm in dynamic environments.

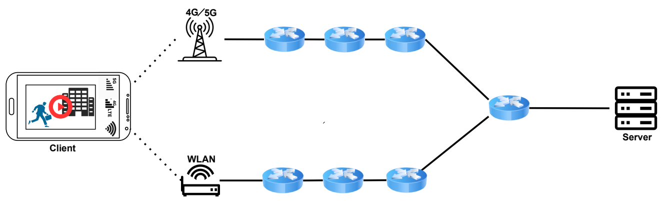


Fig. 10: Simulation Topology.

TABLE I: Simulation Parameters

Path Type	Path Parameters			
	BW (Mbps)	RTT (ms)	Jitter (ms)	Packet Loss Rate (%)
5G	1100	27	± 7	0.1
4G	140	30	± 5	0.1
WLAN	30	20	± 10	0.7

V. Performance Evaluation

In this section, the performance of LSMeta-DAMS is compared with state-of-the-art MP schedulers.

A. Experiment Setup

The MPQUIC implementation in this study is based on the QUIC implementation within mpquic-go [15]. Furthermore, we develop both model-based and learning-based MP schedulers for comparison purposes. minRTT and RR are the non-learning or model-based algorithms, while LSMeta-DAMS, DAMS, Meta-DQN, DQN, ReLes, and Peekaboo are the learning-based schedulers for comparison.

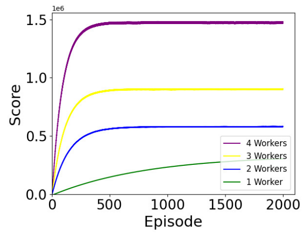
minRTT, RR, ReLes, and Peekaboo are algorithms proposed based on MPTCP. We modified them for our application, specifically adapting them to the MPQUIC framework. On the other hand, LSMeta-DAMS, DAMS, Meta-DQN, and DQN algorithms were developed by us. LSMeta-DAMS is proposed as the primary MP scheduler for this work. We implement Meta-DQN based on our hybrid meta-RL architecture to observe how our MP scheduler performs when using DQN instead of MA3C in our design. Additionally, we develop DAMS and DQN to analyze the behavior of the scheduler without considering our hybrid meta-RL architecture.

We build up a multi-homing scenario in which the client and server are connected using two access links, i.e., WiFi and 4G/5G (Fig. 10). This setup enables us to adjust the RTT ratio of paths and define the client's mobility pattern. In our emulated experiments, we use network traces along with statistical values, aiming for a controlled yet realistic evaluation. Mininet is utilized to emulate the environment, and the characteristics of each network path are shown in Table I [30], [31].

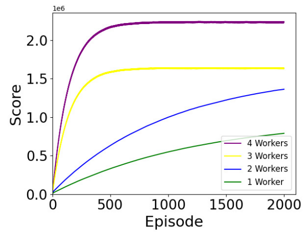
At the application layer, we conduct both bulk transfer and video streaming to evaluate the aggregation capability of the MP schedulers. In the case of bulk transfers, each experiment initiates an HTTP GET request for files of different sizes,

TABLE II: Hyperparameters of the Learning Model

Hyperparameter	Value
Discounted Factor (γ)	0.99
Learning Rate (α)	0.001
$\epsilon_{\max}$	1
$\epsilon_{\min}$	0.01
ϵ_{decay}	0.98

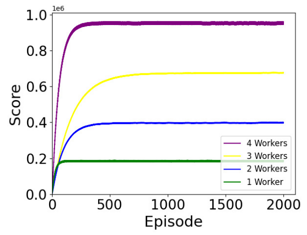


(a) WiFi RTT 20ms, LTE RTT 30ms.

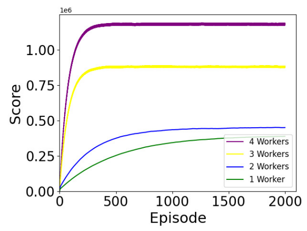


(b) WiFi RTT 20ms, 5G RTT 27ms.

Fig. 11: The Impact of worker count on the performance of LSMeta-DAMS while using meta-models.



(a) WiFi RTT 20ms, LTE RTT 30ms.



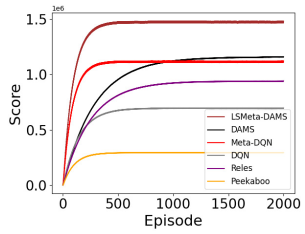
(b) WiFi RTT 20ms, 5G RTT 27ms.

Fig. 12: The Impact of worker count on the performance of LSMeta-DAMS without using meta-models.

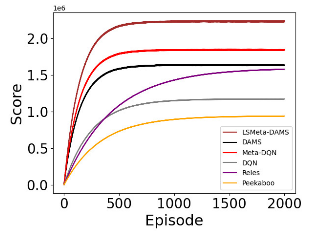
ranging from 256 KB to 256 MB, to assess the performance of the MP schedulers. For video streaming, we utilize a video file containing I, P, and B frames. To extract these video frames and do video quality assessments, we use the free and open-source software project called FFmpeg [40], which provides a collection of libraries and programs for handling multimedia data. We use MPEG-DASH as the adaptive bitrate (ABR) streaming protocol in our video streaming experiments.

B. LSMeta-DAMS Configuration

LSMeta-DAMS's learning components are implemented using keras-rl, a popular deep reinforcement learning library. Based upon the ranges of network conditions covered by the meta-models for each path, we implemented that packet loss rates can range between $[0, 0.5)\%$, $[0.5, 0.75)\%$, and $[0.75, 1)\%$; the mean RTT can be between $[0, 50)$ ms, $[50, 100)$ ms, and $[100, 200]$ ms; and the ratio between deviations and mean RTTs can be $[0, 10)\%$, $[10, 30)\%$, and $[30, 50]\%$. The combination of 27 coarse-grained states can therefore be achieved by a single path [31]. In the case of two paths, there are a total of $3^3 \times 3^3$ different coarse-grained states that can be derived from the potential combinations. Thus, there are 729 meta-models in total. CSV files are routinely used to record online experiences, while Hierarchical Data Format version 5 (HDF5) files are used to store neural networks representing meta-models [31]. Hyperparameters of our learning model are shown in Table II. The discounted factor (γ) in Reinforcement Learning indicates how much an agent values future rewards as part of decision-making. It balances the trade-off between immediate and long-term gains, promoting temporal credit assignment, stability in learning, and exploration in dynamic environments. The choice of γ influences the agent's behavior, with extremes ($\gamma = 0$ or $\gamma = 1$) representing myopic or farsighted decision-making, respectively. Epsilon (ϵ) is utilized in reinforcement learning to address the exploration-exploitation trade-off. Implemented through epsilon-greedy strategies, it governs the agent's decision-making by balancing the exploration of new actions and the exploitation of known actions. Adjusting the value of epsilon enables the agent to adapt its behavior over time, promoting exploration in the early stages of learning and shifting towards exploitation as it gains more knowledge about the environment.



(a) WiFi RTT 20ms, LTE RTT 30ms.



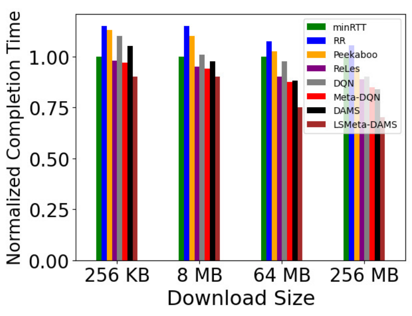
(b) WiFi RTT 20ms, 5G RTT 27ms.

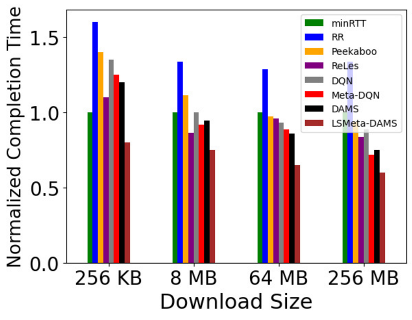
Fig. 13: Score of different MP schedulers.

C. Performance Analysis

*1) **Performance of the learning-based algorithms**:* In this section, our primary objectives are twofold: first, to assess the performance of LSMeta-DAMS using different numbers of workers in its learning algorithm, both with and without the use of meta-models. Second, to compare the performance of learning-based algorithms among themselves, using different network settings.

As shown in Fig. 11a and Fig. 11b, LSMeta-DAMS exhibits improved performance with an increasing number of workers. MA3C's parallel execution allows multiple workers to explore diverse network paths concurrently, facilitating faster exploration, efficient resource utilization, and scalability. The asynchronous nature of MA3C enables independent interactions with the environment, preventing bottlenecks and contributing to quicker learning and adaptation. In the context of managing multiple network paths, the collective knowledge gained from parallel exploration enhances LSMeta-DAMS's decision-making, resulting in a more robust and adaptable to highly dynamic network conditions. This holds true even when no meta-models are present, as illustrated in Fig. 12a. Meta-models accelerate and enhance the convergence accuracy of the online learning model towards the optimal policy. Consequently, the score values in Fig. 11 are higher than those in Fig. 12, showcasing the positive impact of meta-models on performance. The term "score" in this context represents the cumulative reward accumulated by the agent during its learning process, reflecting the total utility achieved by


(a) WiFi RTT 20ms, LTE RTT 30ms.


(b) WiFi RTT 20ms, LTE RTT 150ms.

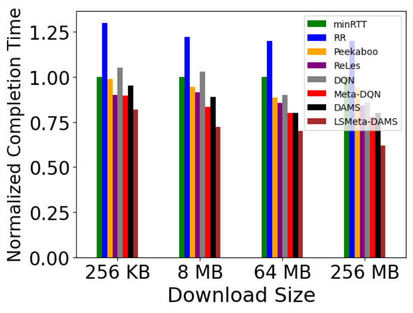
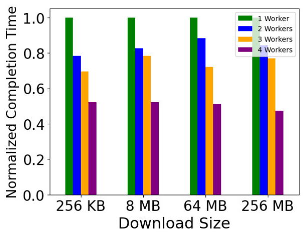

(c) WiFi RTT 20ms, 5G RTT 27ms.

Fig. 14: Performance of LSMeta-DAMS and other schedulers in downloading files of varying sizes.


(a) WiFi RTT 20ms, LTE RTT 30ms.

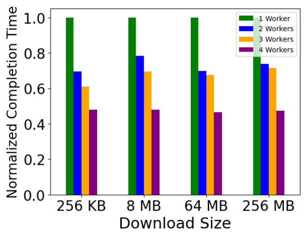

(b) WiFi RTT 20ms, 5G RTT 27ms.

Fig. 15: Performance comparison of LSMeta-DAMS and other schedulers in downloading files with varying sizes, considering different path combinations and worker numbers within the LSMeta-DAMS framework.

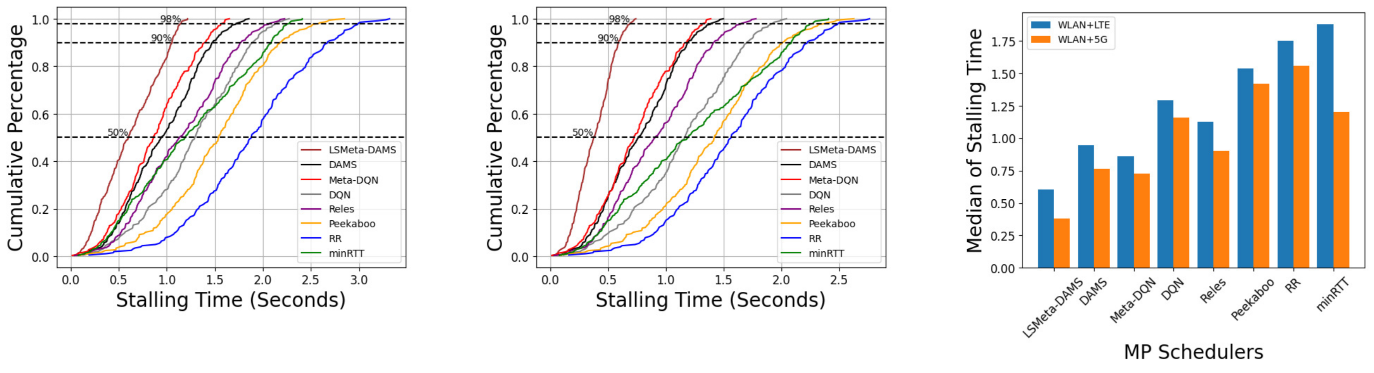
LSMeta-DAMS over multiple episodes of interaction with the environment. It is directly tied to the optimization objectives of our framework, which aim to minimize stalling time, optimize video quality, and maintain smooth playback.

In Fig. 13, we compare the performance of LSMeta-DAMS with other learning-based algorithms under different path configurations. It is worth mentioning that we consider LSMeta-DAMS with four workers in these comparisons. As observed, LSMeta-DAMS outperforms other algorithms in both scenarios, thanks to its hybrid meta-RL architecture and MA3C. However, the performance of Meta-DQN is noteworthy as well, indicating that even if we employ DQN in our proposed hybrid meta-RL MP scheduler framework, it can still outperform other algorithms. Additionally, DAMS, our algorithm without using meta-models, demonstrates a good performance, closely approaching the performance of Meta-DQN.

2) ***Comparison of download time for different RTTs:*** We test the completion (download) time of the schedulers on a variety of files of various sizes, which results are illustrated in Fig. 14a, and Fig. 14b. The term ”completion time” here refers to the total time required to download or transmit a video chunk over the multipath network. To facilitate a fair comparison across different network conditions, we use a regularized or normalized completion time, defined as the ratio of the actual completion time to the baseline completion time of a reference scheduler (minRTT in this case). We examine two RTT combinations to demonstrate network heterogeneity. Compared to minRTT, LSMeta-DAMS increases download speeds by approximately 30% when the RTT values on both paths are similar. LSMeta-DAMS improves download performance by about 40% compared to minRTT as the gap between RTT values widens. Our MP schedulers, LSMeta-DAMS and DQN-based, consistently outperform other MP schedulers in various scenarios. We are able to provide superior performance due to the fact that we include delays and throughput as part of our reward function, in addition to considering RTT, jitter, and path capacity within our state set. LSMeta-DAMS and DQN-based MP schedulers are more adaptable to significant path differences. Nevertheless, LSMeta-DAMS outperforms the DQN-based MP scheduler due to its utilization of A3C within its architecture for learning.

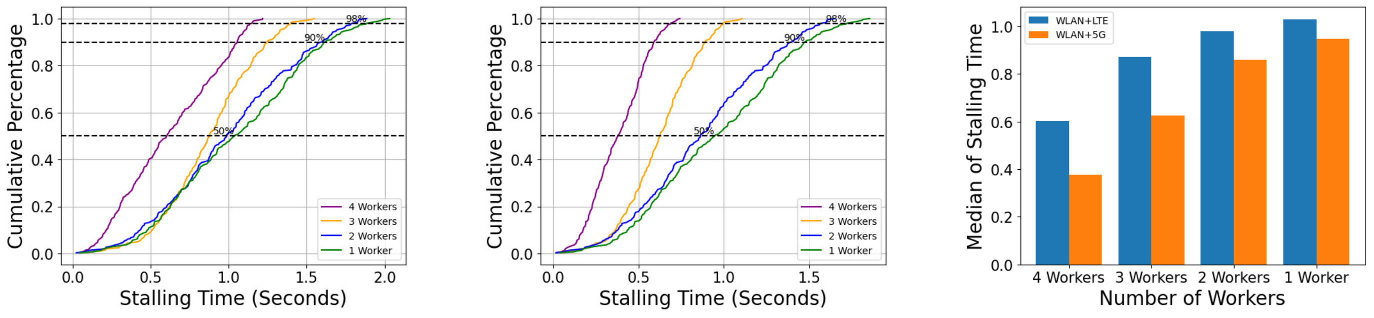
3) ***Comparison of download time for different path combinations:*** Using minRTT as a baseline, we conducted tests on the completion time of various schedulers for different path combinations, such as WLAN/4G and WLAN/5G. Completion times are normalized with respect to the completion time of minRTT. In Fig. 14a, where WLAN and 4G are used as network paths, LSMeta-DAMS demonstrates an approximate 30% increase in download speeds compared to minRTT. Furthermore, when considering WLAN and 5G as network paths, LSMeta-DAMS exhibits a 38% improvement in download performance compared to minRTT, as shown in Fig. 14c. LSMeta-DAMS outperforms other MP schedulers, particularly Meta-DQN and DAMS, which share the same architecture as LSMeta-DAMS. This superiority is due to the integration of hybrid meta-RL and MA3C within its learning architecture. Another contributing factor to its superior performance is the inclusion of delays and throughput as part of the reward function.

4) ***Download Time Comparison with Varying Worker Numbers in LSMeta-DAMS Framework for Different Path Combinations:*** As illustrated in Fig. 15, in both scenarios, an increase in the number of workers in LSMeta-DAMS's learning algorithm MA3C corresponds to a reduction in completion time. This improvement can be attributed to the inherent parallelization capabilities of MA3C. With a higher number of workers, each worker can independently explore and exploit the environment, sending its feedback to the global network. Thus, the global network has access to more information in each time slot compared to other algorithms, enabling it to



(a) WiFi RTT 20ms, LTE RTT 30ms. (b) WiFi RTT 20ms, 5G RTT 27ms. (c) Median Values of Stalling Time.

Fig. 16: Total stalling time of different MP schedulers.



(a) WiFi RTT 20ms, LTE RTT 30ms. (b) WiFi RTT 20ms, 5G RTT 27ms. (c) Median Values of Stalling Time.

Fig. 17: Total stalling time of LSMeta-DAMS.

converge to the optimal policy faster and more accurately.

5) ***Stalling Time:*** As mentioned in section II-C, stalling time refers to pauses or buffering interruptions during playback. It represents the gap between the end of the playback of one chunk (C_i) and the complete arrival of the next chunk (C_{i+1}). If the next chunk arrives before the current chunk's playback concludes, there is no stalling time. However, if the playback of C_i finishes and the C_{i+1} is not fully received, the time gap until the complete arrival of C_{i+1} is considered as the stalling time. In simpler terms, stalling time measures the delay or waiting period between consecutive chunks of a video. For each MP scheduler, we conduct 100 experiments and present the cumulative distribution function (CDF) of the total stalling time during the whole session, reflecting user experience, in both Fig. 16 and Fig. 17.

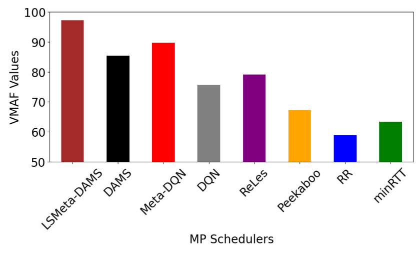
As illustrated in Fig. 16, LSMeta-DAMS exhibits the least stalling time compared to other MP schedulers in both scenarios. This superiority stems from LSMeta-DAMS's ability to rapidly and accurately schedule packets across available paths, not only minimizing the risk of buffer depletion but also proactively fetching and buffering content.

In Fig. 17, We aim to demonstrate the impact of the number of workers on the performance of LSMeta-DAMS. It is evident that having more workers can significantly enhance the performance of LSMeta-DAMS. As mentioned before, with a higher number of workers, each worker is able to independently explore and exploit the environment, resulting in faster and more accurate convergence to the best scheduling policy.

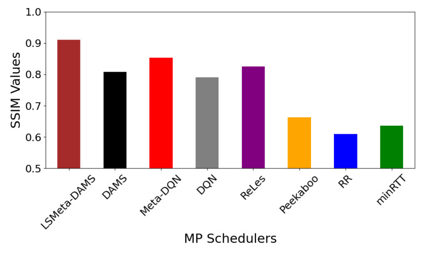
6) ***Video Quality Assessment:*** In assessing the impact of our proposed MP packet scheduler on live video streaming, we deploy three widely used metrics: Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM), and Video Multimethod Assessment Fusion (VMAF). PSNR, a conventional metric, provides a straightforward measure of fidelity between the original and received videos through signal power ratios. Calculated as the ratio between peak signal power and mean squared error (MSE), a higher PSNR value indicates lower distortion and superior video quality [41]. Acceptable range for PSNR is from 30 dB to 50 dB [42]. SSIM introduces perceptual accuracy by incorporating structural information, offering a scale from 0 to 1, where values exceeding 0.9 are often considered acceptable [43]. VMAF, developed by Netflix, employs machine learning for content-dependent assessments, providing scores on a 0 to 100 scale [44].

Enhancing video quality stands as a crucial objective for quality metrics. However, widely employed metrics such as PSNR and SSIM, despite their ease of calculation, fall short in capturing the subjective perceptions of the human eye. As a result, they offer limited practical value in real-world applications. However, to thoroughly assess the visual quality of the received video using LSMeta-DAMS in comparison to other MP schedulers, we calculate all three metrics using existing libraries in FFmpeg. The video quality of different MP schedulers for various path combinations is illustrated in Fig. 18 and Fig. 19.

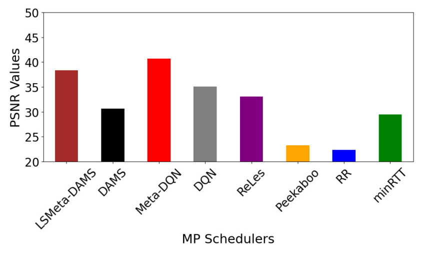
As illustrated in Fig. 18a and Fig. 19a, the VMAF value for LSMeta-DAMS surpasses that of other MP schedulers.



(a) VMAF

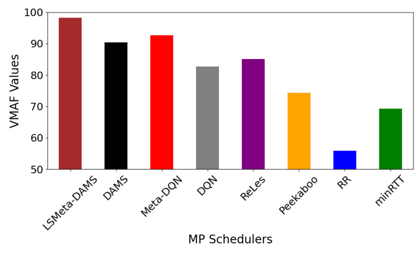


(b) SSIM

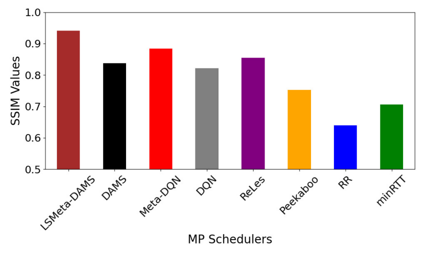


(c) PSNR

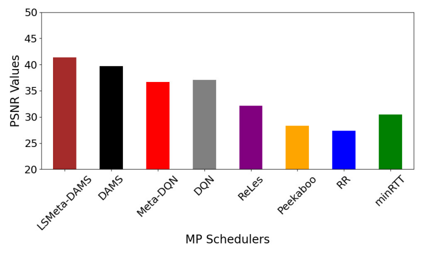
Fig. 18: Video quality assessment considering three metrics, including PSNR, SSIM, and VMAF (WLAN+LTE).



(a) VMAF



(b) SSIM



(c) PSNR

Fig. 19: Video quality assessment considering three metrics, including PSNR, SSIM, and VMAF (WLAN+5G).

LSMeta-DAMS also excels over other MP schedulers in terms of PSNR and SSIM metrics, except in Fig. 18c. Nevertheless, the VMAF value holds particular significance for us in this context, as our primary goal is to ensure the quality of the received video aligns with human perception. This superior performance in terms of speed and accuracy is attributed to the hybrid meta-RL architecture of LSMeta-DAMS and the utilization of MA3C as the learning algorithm. Additionally, LSMeta-DAMS incorporates different video frames (IPB frames) and layers (base layer and enhancement layer) in its design. The consideration of diverse frame types and the incorporation of parameters that contribute to reduced stalling time in our reward design are other key factors contributing to LSMeta-DAMS's exceptional performance.

VI. Conclusion And Further Research Issues

In conclusion, this paper introduces LSMeta-DAMS, a learning-based multipath scheduler designed for MP live streaming. LSMeta-DAMS utilizes a hybrid meta-reinforcement learning architecture with offline and online phases. Unique features, such as prioritizing I-frame packets, video chunk and frame management, and an enhanced asynchronous training algorithm based on A3C (MA3C), collectively contribute to superior performance in learning, accuracy, and overall video streaming quality. Practical implementation and trace-driven emulations using mpquic-go, TensorFlow, and keras-rl validate LSMeta-DAMS's effectiveness, showcasing its potential to significantly enhance mobile video streaming experiences.

Two promising future directions stem from this work. Firstly, we propose developing a coupled congestion control algorithm tailored for the multipath scheduler, aiming to synchronize congestion control mechanisms and optimize MP video streaming performance. Secondly, we envision creating a dedicated multipath dynamic adaptive streaming over HTTP (MP DASH) algorithm addressing unique challenges in multipath video streaming.

References

[1] A. Sepahi, L. Cai, W. Yang, and J. Pan, "Meta-DAMS: Delay-aware multipath scheduler using hybrid meta-reinforcement learning," in *2023 IEEE 98th Vehicular Technology Conference (VTC2023-Fall)*, pp. 1–5, 2023.

[2] V. G. Cerf and R. E. Kahn, "A protocol for packet network intercommunication," *IEEE Transactions on Communications*, vol. 22, pp. 637–648, 1974.

[3] A. Langley et al., "The QUIC transport protocol," *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017.

[4] D. Bhat, A. Rizk, and M. Zink, "Not so QUIC: A performance study of DASH over QUIC," in *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV'17, (New York, NY, USA), pp. 13–18, Association for Computing Machinery, 2017.

[5] A. Ford, C. Raiciu, M. J. Handley, and O. Bonaventure, "TCP extensions for multipath operation with multiple addresses." RFC 6824, 2013.

[6] Q. De Coninck and O. Bonaventure, "Multipath QUIC: Design and evaluation," in *Proceedings of the 13th International Conference on Emerging Networking Experiments and Technologies*, pp. 160–166, 2017.

[7] W. Yang, L. Cai, S. Shu, and J. Pan, "Scheduler design for mobility-aware multipath QUIC," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, pp. 2849–2854, 2022.

[8] W. Yang, L. Cai, S. Shu, J. Pan, and A. Sepahi, "Mams: Mobility-aware multipath scheduler for mpquic," *IEEE/ACM Transactions on Networking*, pp. 1–16, 2024.

[9] H. Wu, G. Caso, S. F. Oliveira, Ö. Alay, and A. Brunström, "Multipath scheduling for 5G networks: Evaluation and outlook," *IEEE Communications Magazine*, vol. 59, pp. 44–50, 2021.

[10] V. Mnih et al., "Playing Atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[11] P. Bestagini, S. Milani, M. Tagliasacchi, and S. Tubaro, "Codec and GOP identification in double compressed videos," *IEEE Transactions on Image Processing*, vol. 25, pp. 2298–2310, 2016.

[12] J.-S. Lee et al., "Subjective evaluation of scalable video coding for content distribution," in *Proceedings of the 18th ACM International Conference on Multimedia*, MM '10, (New York, NY, USA), pp. 65–72, Association for Computing Machinery, 2010.

[13] I. Sodagar, "The MPEG-DASH standard for multimedia streaming over the Internet," *IEEE MultiMedia*, vol. 18, no. 4, pp. 62–67, 2011.

[14] V. Mnih et al., "Asynchronous methods for deep reinforcement learning," in *33rd ICML*, 2016.

[15] D. C. Quentin, "MPQUIC-go." https://github.com/qdeconinck/mp-quic/, 2019.

[16] M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[17] M. Plappert, "Keras-rl." https://github.com/keras-rl/keras-rl, 2016.

[18] H. Wu, S. Ferlin, G. Caso, . Alay, and A. Brunstrom, "A survey on multipath transport protocols towards 5G access traffic steering, switching and splitting," *IEEE Access*, vol. 9, pp. 164417–164439, 2021.

[19] T. W. do Prado Paiva, S. Ferlin, A. Brunstrom, O. Alay, and B. Y. L. Kimura, "A first look at adaptive video streaming over multipath QUIC with shared bottleneck detection," *Proceedings of the 14th Conference on ACM Multimedia Systems*, 2023.

[20] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, 2007.

[21] J. Chen, C. Kao, and Y. Lin, "Introduction to H.264 advanced video coding," *Asia and South Pacific Conference on Design Automation, 2006.*

[22] S. Yu and N. Wang, "Video stalling detection method," *IOP Conference Series: Materials Science and Engineering*, vol. 715, no. 1, pp. 12–36, 2020.

[23] A. Biernacki and K. Tutschku, "Performance of HTTP video streaming under different network conditions," *Multimedia Tools and Applications*, vol. 72, pp. 1143–1166, 2014.

[24] H. F. Harlow, "The formation of learning sets," *Psychological Review*, vol. 56, no. 1, pp. 51–65, 1949.

[25] R. Hattori et al., "Meta-reinforcement learning via orbitofrontal cortex," *Nature Neuroscience*, 12 2023.

[26] S. Ferlin, Ö. Alay, O. Mehani, and R. Boreli, "BLEST: Blocking estimation-based MPTCP scheduler for heterogeneous networks," in *2016 IFIP Networking Conference (IFIP Networking) and Workshops*, pp. 431–439, IEEE, 2016.

[27] Y.-s. Lim, E. M. Nahum, D. Towsley, and R. J. Gibbens, "ECF: An MPTCP path scheduler to manage heterogeneous paths," in *Proceedings of the 13th International Conference on Emerging Networking Experiments and Technologies*, pp. 147–159, 2017.

[28] H. Wu, . Alay, A. Brunstrom, S. Ferlin, and G. Caso, "Peekaboo: Learning-based multipath scheduling for dynamic heterogeneous environments," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2295–2310, 2020.

[29] S. Lee and J. Yoo, "Reinforcement learning based multipath QUIC scheduler for multimedia streaming," *Sensors*, vol. 22, no. 17, 2022.

[30] H. Zhang, W. Li, S. Gao, X. Wang, and B. Ye, "ReLeS: A neural adaptive multipath scheduler based on deep reinforcement learning," in *IEEE INFOCOM 2019*, 2019.

[31] H. Wu et al., "FALCON: Fast and accurate multipath scheduling using offline and online learning," *arXiv preprint arXiv:2201.08969*, 2022.

[32] X. Zhao, B. Liu, X. Jiang, W. Tang, and W. Dou, "Smart decision for device selection in D2D-assisted multipath video transmission network," *Expert Systems*, vol. 40, 2022.

[33] D. Jurca and P. Frossard, "Video packet selection and scheduling for multipath streaming," *IEEE Transactions on Multimedia*, vol. 9, no. 3, pp. 629–641, 2007.

[34] M. R. Garey and D. S. Johnson, *Computers and intractability – A guide to the theory of NP-completeness*. New York: W. H. Freeman, 23rd ed., 2002.

[35] V. H. Pong et al., "Offline meta-reinforcement learning with online self-supervision," in *International Conference on Machine Learning*, pp. 17811–17829, PMLR, 2022.

[36] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Comput. Surv.*, vol. 53, no. 3, 2020.

[37] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[38] T. Tieleman, G. Hinton, *et al.*, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural Networks for Machine Learning*, vol. 4, no. 2, pp. 26–31, 2012.

[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[40] F. Project, "FFmpeg: A complete, cross-platform solution to record, convert, and stream audio and video." https://ffmpeg.org/, 2023.

[41] F. A. Fardo, V. H. Conforto, F. C. de Oliveira, and P. S. Rodrigues, "A formal evaluation of PSNR as quality measurement parameter for image segmentation algorithms," 2016.

[42] J. Erfurt et al., "A study of the perceptually weighted peak signal-to-noise ratio (WPSNR) for image compression," in *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 2339–2343, 2019.

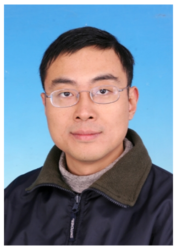
[43] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[44] R. Rassool, "VMAF reproducibility: Validating a perceptual practical video quality metric," in *2017 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pp. 1–2, 2017.

Amir Sepahi is a Ph.D. Candidate in the Department of Electrical & Computer Engineering at the University of Victoria, British Columbia, Canada. He obtained his M.Sc. in Communication Networks from Isfahan University of Technology in 2019. His research interests include Wireless Communications, Internet Protocols, Artificial Intelligence, and Network Security.

Lin Cai (S'00-M'06-SM'10-F'20) has been with the Department of Electrical & Computer Engineering at the University of Victoria since 2005, and she is currently a Professor. She is an NSERC E.W.R. Steacie Memorial Fellow, an Engineering Institute of Canada Fellow, a Canadian Academy of Engineering Fellow, a Royal Society of Canada Fellow, and an IEEE Fellow. Her research interests span several areas in communications and networking, with a focus on network protocol and architecture design supporting emerging multimedia traffic and the Internet of Things. She has been elected to serve the board of the IEEE Vehicular Technology Society, 2019 - 2024, and as its VP in Mobile Radio. She has been a Board Member of IEEE Women in Engineering (2022-24) and IEEE Communications Society (2024-2026). She has served as an Associate Editor-in-Chief for IEEE Transactions on Vehicular Technology, and as a Distinguished Lecturer of the IEEE VTS Society and the IEEE Communications Society.

Wenjun Yang is currently a Postdoctoral fellow in the Department of Computer Science at the University of Victoria, British Columbia, Canada. His research interests include multipath QUIC/TCP, video streaming, deep reinforcement learning, and the next generation of network architecture. He is a Graduate Student Member of IEEE.



Jianping Pan is a Professor of Computer Science at the University of Victoria, British Columbia, Canada. He received his Bachelor's and PhD degrees in computer science from Southeast University, Nanjing, Jiangsu, China, and he did his postdoctoral research at the University of Waterloo, Ontario, Canada. He also worked at Fujitsu Labs and NTT Labs. His area of specialization is computer networks and distributed systems, and his current research interests include protocols for advanced networking, performance analysis of networked systems, and applied network security. He received IEICE Best Paper Award in 2009, Telecommunications Advancement Foundation's Telesys Award in 2010, WCSP 2011 Best Paper Award, IEEE Globecom 2011 Best Paper Award, JSPS Invitation Fellowship in 2012, IEEE ICC 2013 Best Paper Award, NSERC DAS Award in 2016, IEEE ICDCS 2021 Best Poster Award and DND/NSERC DGS Award in 2021, and has been serving on the technical program committees of major computer communications and networking conferences including IEEE INFOCOM, ICC, Globecom, WCNC and CCNC. He was the Ad Hoc and Sensor Networking Symposium Co-Chair of IEEE Globecom 2012 and an Associate Editor of IEEE Transactions on Vehicular Technology. He is a senior member of the ACM and a Fellow of the IEEE.