EdAR: An Experience-Driven Multipath Scheduler for Seamless Handoff in Mobile Networks

Jiangping Han, *Member, IEEE*, Kaiping Xue, *Senior Member, IEEE*, Jian Li, *Member, IEEE*, Rui Zhuang, *Graduate Student Member, IEEE*, Ruidong Li, *Senior Member, IEEE*, Ruozhou Yu, *Senior Member, IEEE*, Guoliang Xue, *Fellow, IEEE*, and Qibin Sun, *Fellow, IEEE*

***Abstract*— Multipath TCP (MPTCP) improves the bandwidth utilization in wireless network scenarios, since it can simultaneously utilize multiple interfaces for data transmission. However, with the fast growth of mobile devices and applications, link interruptions caused by handoffs still lead to drastic performance degradation in such scenarios. Typically, a series of packet losses on part of the links will block the transmission of the entire connection when handoff occurs. This paper proposes an Experience-driven Adaptive Redundant packet scheduler (EdAR) for MPTCP, aiming at achieving seamless handoffs in mobile networks. EdAR enables flexibly scheduling redundant packets with an experience-driven learning-based approach in the face of drastic network environment changes for multipath performance enhancement. To enable accurate learning and prediction, both the network environment and the best course of actions are jointly learned via a Deep Reinforcement Learning (DRL) agent, which we design with a hybrid structure to deal with the complexity of system states. Furthermore, both offline and online learning are utilized to allow the agent to adapt to different and changing network environments. Evaluation results show that EdAR outperforms the state-of-the-art MPTCP schedulers in most network scenarios. Specifically in mobile networks with frequent handoffs, EdAR brings 2× improvement in terms of the overall goodput.**

***Index Terms*— MPTCP, scheduler, handoff, experience-driven, deep reinforcement learning.**

I. Introduction

With the development of wireless networks, mobile devices are growing rapidly and have become the mainstream in modern networks [1], [2], [3], [4]. At the same time, the increasing demands of users and applications lead to a further rise in requirement for bandwidth and robustness in wireless scenarios [5], [6]. To meet the demands, multipath transmission protocols, such as Multipath TCP (MPTCP) [7], [8], have become a major trend in today's mobile wireless networks with the prevalence of multi-homed terminals. By dividing a conventional TCP flow into multiple subflows, MPTCP can make use of as many connected interfaces of a device as possible to achieve bandwidth aggregation and improve quality-of-service (QoS) for users [9], [10], [11]. Ideally, it can aggregate bandwidth and overcome transmission instability on a single path to improve the reliability of mobile communications [12], [13], [14].

However, in wireless mobile scenarios, handoffs caused by user mobility is becoming the main factor affecting network transmission [15], [16]. MPTCP still faces challenges in such scenarios, where the unpredictable degradation of a single path may severely degrade the performance of the whole MPTCP connection [17], [18]. Specifically, existing works have shown that MPTCP does not perform well in the network scenarios where the path conditions change fast. The asymmetric path conditions (such as different packet loss and delay) can make an unexpected performance degradation of the entire MPTCP connection. To deal with this issue, MPTCP utilizes multipath scheduling schemes to allocate different packets on subflows and improve performance under various scenarios [19], [20], [21]. The key idea is to adjust the scheduling strategy based on the estimation of subflow conditions and out-of-order packets, reducing head-of-line (HoL) blocking caused by path asymmetry. However, most previous works do not consider the frequent handoffs, hence they cannot handle the degradation in mobile scenarios. If a large number of consecutive packet losses occur, the sender still needs to wait until several timeouts to retransmit the lost data, thus blocking the entire connection and causing performance degradation. This is why MPTCP's performance is not as superimposed as expected, or sometimes even worse than a TCP flow on the best path [22], [23].

Fortunately, the use of multiple subflows makes MPTCP possible to handle packet loss or link interruption in parallel on different subflows to improve robustness. One approach to improve robustness and ensure low latency is to always transmit the same redundant copies of packets on different subflows. Therefore, the packet loss on one subflow will not affect the transmission of the entire connection. However, it causes a duplicate packet delivery and a waste of bandwidth

Manuscript received 19 July 2022; revised 8 December 2022 and 9 February 2023; accepted 12 February 2023. Date of publication 24 February 2023; date of current version 11 October 2023. The work of Jiangping Han, Kaiping Xue, Jian Li, and Rui Zhuang was supported in part by the National Natural Science Foundation of China under Grant 61972371, in part by the Youth Innovation Promotion Association of Chinese Academy of Sciences (CAS) under Grant Y202093, and in part by the Fundamental Research Funds for the Central Universities. The work of Qibin Sun was support in part by the NSFC under Grant U19B2023. The associate editor coordinating the review of this article and approving it for publication was J. Liu. *(Corresponding author: Kaiping Xue.)*

Jiangping Han, Kaiping Xue, Jian Li, Rui Zhuang, and Qibin Sun are with the School of Cyber Science and Technology, University of Science and Technology of China, Hefei 230027, China (e-mail: kpxue@ustc.edu.cn).

Ruidong Li is with the National Institute of Information and Communications Technology, Kanazawa University, Tokyo 184-0015, Japan.

Ruozhou Yu is with the Department of Computer Science, North Carolina State University, Raleigh, NC 27606 USA.

Guoliang Xue is with the School of Computing and Augmented Intelligence, Arizona State University, Tempe, AZ 85287 USA.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TWC.2023.3246082.

Digital Object Identifier 10.1109/TWC.2023.3246082

resources when the network condition is good and stable. To achieve both efficiency and robustness, a scheduler should be well-designed to smoothly change between sending redundant copies of packets and sending new packets. It also needs the ability to accurately decide when to send and not send redundant copies of packets according to real-time path conditions in wireless mobile networks where the network condition changes fast. Due to the dynamic network environments and the complexity of mobile scenarios, a traditional model-based approach can hardly be applied to all situations nor adapt to the rapid changes in mobile networks. To overcome the issue of unpredictable performance degradation in wireless mobile networks, we consider a learning-based approach to learn new knowledge from past experiences and constantly update decisions to better adapt to the changing network environments.

In this paper, we propose an Experience-driven Adaptive Redundant MPTCP packet scheduler (EdAR) that utilizes flexible redundant packet scheduling to overcome the transmission instability in changing network scenarios. To satisfy this requirement, EdAR conducts a redundant buffer linking to the send buffer and subflows for smoothly switching between redundant and normal transmissions. Further, it utilizes Deep Reinforcement Learning (DRL) to learn an optimal policy, taking into account the hybrid network state information including current connection states and alterable subflow states with historical information. EdAR also formulates multiple measurable network performance metrics including goodput, bandwidth utilization, and out-of-order packets to enable accurate decisions. To address the cold start problem of the DRL agent and asynchronism in MPTCP transmission, EdAR includes both offline learning and asynchronous online learning, where the online learning is decoupled with asynchronous data collection of the sender and receiver to keep low computational overhead and fast online scheduling. The contributions of this paper are summarized as follows:

- We design EdAR, an MPTCP packet scheduler to overcome the poor performance of MPTCP in mobile scenarios. EdAR enables flexible redundant packet scheduling with an experience-driven agent extracting the most appropriate decision accordingly, ensuring both high bandwidth utilization and seamless handoff in wireless networks.
- EdAR takes a multi-objective reward function considering various performance metrics that make different contributions. A special network structure is designed for the EdAR agent that consists of a representation network and a core network, which gives EdAR the ability to deal with the complex relationship of system states and make full use of alterable network interfaces.
- EdAR runs a learning approach offline and updates online to adapt to different scenarios. Based on extensive NS-3 simulation experiments, we show that EdAR has superior performance compared to state-of-the-art schedulers in mobile scenarios, making 2× improvement in terms of goodput during handoff situations.

The rest of the paper is organized as follows. Section II describes the background and related work. Section III describes the problem formulation in handoff situations. Sections IV and V describe the design of EdAR scheduler and learning approaches, respectively. Section VI presents the evaluation results. Finally, Section VII concludes this paper.

II. Background, Related Work and Preliminaries

A. Overview of Multipath Transmission Protocols

Nowadays, most of the end-hosts are equipped with more than one interface, and users expect to be able to use them simultaneously for performance enhancement [24], [25]. MPTCP is one of the most popular multipath transport protocols, which extends TCP to utilize multiple paths simultaneously for bandwidth aggregation as well as ensuring reliable transmission [26]. With the development of MPTCP gradually improving, it has become an important transmission protocol in current mobile networks [27], [28]. While fully compatible with existing network protocols, MPTCP can greatly improve transmission performance, providing higher transmission efficiency and better robustness. However, with the increase of user mobility in wireless networks, it has been shown that MPTCP meets unpredictable degradation when path conditions change fast. For example, Li et al. [3] showed that the performance of MPTCP declines significantly on high-speed rails, where the low efficiency can be attributed to poor adaptability to frequent handoffs under high-speed mobility. While coupled congestion control algorithms [29], [30], [31], [32] can be used to jointly control subflow rates for enhanced migratability, condition changes on one path can still negatively affect the performance of the entire MPTCP connection, such as when increased delay and packet losses happen due to handoffs or link interruptions.

B. Overview of Multipath Packet Scheduling Schemes

To deal with different and changing network environments, MPTCP can utilize multipath packet scheduling schemes that schedule packets on different subflows to improve performance under different scenarios. Existing multipath packet scheduling schemes mainly aim at avoiding out-of-order packets and improving overall throughput [33], [34], [35], [36]. Among them, two basic static schemes are Round-Robin (RR) and minimum RTT first (minRTT) [37]. RR constantly polls subflows and sends packets to aggregate bandwidth, but increases RTT when the RTT gap among subflows is large. MinRTT sends packets first on the available subflow with the minimum RTT to reduce RTT. Compared with the static packet scheduling schemes, ECF [33] and BLEST [34] change the scheduling decision according to the real-time path conditions to enhance transmission performance. The main idea behind them is to keep the buffer blocking under control by deciding whether to send new packets on a subflow or pause a subflow from sending packets. ECF and BLEST all treat the fast subflow (i.e., the subflow with the shortest RTT) as the best subflow, and send new packets to the fastest subflow whenever it becomes available. Otherwise, the slower subflows are only utilized to send new packet when they do not make HoL blocking. Therefore, the out-of-order packets can be reduced in asymmetric scenarios.

To be generically applicable in different and changing network environments, some learning-based schemes, such as ReLeS [35] and Peekaboo [36], are proposed to make scheduling decisions based on different scheduling methods and metrics. ReLeS applies a learning-based approach to teach the multipath scheduler to find the best set of split ratios of packets over subflows. Peekaboo is designed for MPQUIC [38], but the mechanism is also applicable in MPTCP. Peekaboo utilizes a lightweight online learning solution based on Multi-Armed Bandit (MAB) to decide whether and when to pause a subflow from sending packets, so as to reduce out-of-order packets. The use of learning-based approaches makes them more adaptable to the dynamic characteristics of the paths. However, they do not consider the handoff situations in wireless networks.

Besides, there are also some schedulers that use model-based redundant transmissions with different performance metrics. RAVEN [19] mitigates tail latency and network unpredictability for latency-sensitive traffic by using redundant transmissions. To ensure low latency, RAVEN selects the subflow with the shortest RTT to send data. If the confidence interval of the measured RTT is unreliable, RAVEN sends redundant data on other subflows to ensure low latency. The redundant transmission mode is only used for short flows, i.e. when the kernel data queue is smaller than a preset threshold. If the kernel data queue is large, RAVEN considers the flow as non-latency-sensitive and uses the original non-redundant transmission to aggregate bandwidth. As a result, RAVEN greatly improves the delay performance of short delay-sensitive flows, however, it does not provide dynamic adjustment for long flows. AR&P [14] is a scheduler that designs for BBR-based coupled congestion control algorithm (Coupled BBR) with adaptively redundant transmission to improve the throughput in wireless networks. It decides whether to send redundant packets on a subflow in "poor" states based on the bandwidth and delay information detected by Coupled BBR. However, since AR&P utilizes the measurement result from Coupled BBR and is designed for Coupled BBR's pacing mechanism, it does not suitable for all congestion control algorithms as a universal scheduler.

Moreover, for the model-based packet schedulers, they can only make decisions based on a fixed model and current transmission states, for example, the given threshold and the current measured RTT, etc. However, in mobile networks and heterogeneous networks, a fixed model cannot be suitable for all the different network environments. Once the network environment changes, they will fail to adapt to the new network environment. In addition, in mobile networks, there is hysteresis in the current measurements, which can cause bias in future decisions. Using Deep Reinforcement Learning (DRL) can learn the trends of the changing network environment and make more accurate scheduling decisions. In this paper, the proposed EdAR scheduler utilizes flexible redundant scheduling strategies with a DRL-based approach and focuses on the handoff situations in mobile scenarios. By constantly learning from historical experiences, EdAR enables seamless handoff as well as high transmission efficiency in wireless networks.

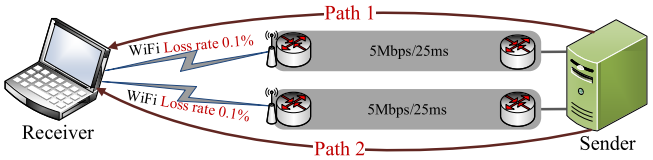


Fig. 1. Test topology under handoff situations.

C. Deep Reinforcement Learning

DRL is a combination of Deep Learning (DL) and Reinforcement Learning (RL), which integrates DL's strong understanding ability in vision and other perceptual problems as well as the decision-making ability of RL. The emergence of DRL makes learning-based technologies practical and can solve complex problems in real scenes. In a general DRL framework, the agent interacts with the environment, observes the environment states $\mathbf{s}_t$, executes strategies a_t, and receives rewards r_t. With feedback reward information from the environment, the agent can find a policy of mapping its states to actions that optimize the cumulative discount reward. Let $\mu_{\boldsymbol{\theta}}$ denote a policy that chooses actions depending on the states, $\mathbf{a}_t = \mu_{\boldsymbol{\theta}}(\mathbf{s}_t)$. With feedback reward information from the environment, DRL learns the optimal policy $\mu^*_{\boldsymbol{\theta}}$ for mapping its states to actions that optimize the cumulative discount reward. Because of its powerful learning ability and adaptability to changing environments, DRL is increasingly being used in network optimization [39], [40], [41], [42].

III. Problem Formulation in Handoff Situations

In mobile networks, the transmission performance is easily affected by frequent handoffs. Packet loss and link interruption of a single path can seriously affect the overall transmission performance. Specifically, lost packets on one subflow will cause a large number of out-of-order packets at the receiver and block the entire MPTCP connection, even when all other subflows do not have packet loss nor performance degradation. As a result, it leads to a throughput degradation in mobile scenarios.

We show the real-time performance of MPTCP using a test topology shown in Fig. 1 to illustrate the performance degradation caused by a single path handoff. The MPTCP connection has two subflows, each of which goes through a path with bandwidth of 5 Mbps, delay of 25 ms, and loss rate of 0.1%. During the transmission, path 2 suffers handoffs at 20-40s and 60-80s. Fig. 2 shows the real-time goodput of MPTCP using minRTT and ReMP [37]. Different from throughput, goodput is the actual receiving rate at the receiver, which expects the redundant and lost packets. MinRTT is the default scheduler of MPTCP, which always schedules new packets on any available subflow to aggregate bandwidth. ReMP utilizes a redundant multipath scheduling scheme, which schedules redundant packets on different subflows to ensure robustness. As shown in Fig. 2, goodput is the actual throughput at a receiver excluding redundant packets and lost packets. TCP 1 and TCP 2 denote the goodput of a TCP flow running alone

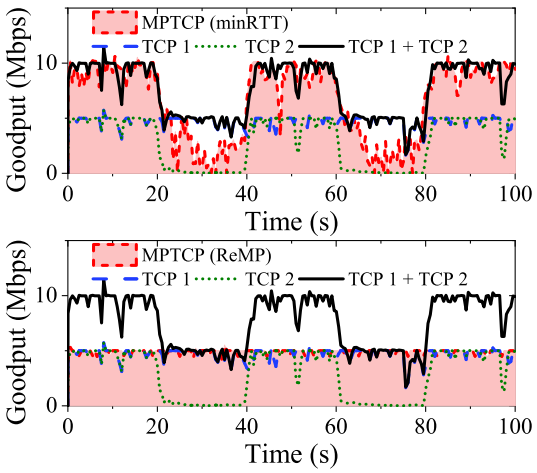


Fig. 2. Real-time goodput of minRTT and ReMP schedulers.

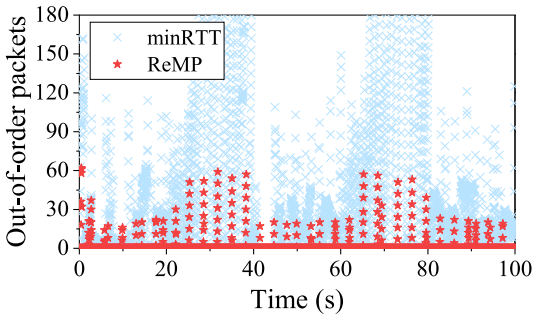


Fig. 3. Real-time out-of-order packets of minRTT and ReMP schedulers.

on path 1 and path 2, respectively. "TCP 1 + TCP 2" denotes the cumulative goodput of TCP 1 and TCP 2.

During the transmission, TCP 1 always delivers a goodput of 5 Mbps. The goodput of TCP 2 decreases to close to 0 Mbps when handoff happens. MPTCP with minRTT provides high bandwidth utilization under stable situations (that is, both two paths are not in handoff). But the goodput of MPTCP with minRTT decreases significantly at 20-40s and 60-80s when one of the subflows suffers handoff situations, which is less than the goodput of a TCP flow on the best paths. Different from minRTT, ReMP sends redundant copies of packets on each subflow to provide high robustness in mobile network scenarios. ReMP provides at least the same goodput as the best TCP flow when handoff happens. However, sending redundant packets causes a waste of bandwidth in stable situations, where the goodput of a MPTCP connection utilizing two paths is only 50% of the cumulative goodput of TCP 1 and TCP 2.

Fig. 3 shows the out-of-order packets at the MPTCP receiver during 0 to 100 s. MPTCP with minRTT causes a much larger number of out-of-order packets during handoffs compared to MPTCP with ReMP, and the out-of-order packets even fully occupy the receive buffer and cause HoL blocking, severely affecting transmission performance. In contrast, MPTCP with ReMP has maintained a very low number of out-of-order packets. Even when handoff happens, there is only a small increase in the number of out-of-order packets.

Therefore, to provide high bandwidth utilization as well as seamless handoff performance, we propose a scheduler to smartly send redundant packets according to the real-time path and connection conditions, based on several key performance indicators include goodput, bandwidth utilization, and out-of-order packets. Due to complexity in the network state and dynamic network conditions, it is hard for a model-based approach to distinguish between different path conditions, such as random loss, congestion loss, and handoff situations to make accurate decisions, especially in mobile scenarios where path conditions change fast.

DRL has been proposed in existing transmission protocols, such as congestion control [43], [44], routing [45], and scheduling [46] algorithms. A DRL-based approach can make decisions according to real-time path conditions and update strategies when the network status changes. Therefore, it can be suitable for dynamic network situations. However, previous scheduling schemes do not consider the impact of mobile network scenarios and handoff situations, and therefore they do not perform well in dynamic mobile scenarios. In this paper, we study this issue and design an experience-driven multipath scheduler, named EdAR, to achieve seamless handoff with high QoS for MPTCP transmissions.

IV. Design of EdAR

In this section, we present the design of Experience-driven Adaptive Redundant packet scheduler (EdAR) for MPTCP.

A. EdAR Scheduling Strategies

EdAR utilizes a flexible redundant packet scheduling scheme to provide transmission robustness with high efficiency guarantee, especially during handoff situations in mobile networks. The advantage of sending redundant packets is considered as two aspects. First, sending redundant packets is used to improve transmission performance when the path conditions are far from optimal. When handoff occurs on a subflow and causes packet loss, sending redundant packets on other subflows can prevent transmission from being blocked by lost packets immediately. Second, sending redundant packets is used to always keep the activity of subflow. That is to say, a subflow needs to always send packets to trace path states in mobile scenarios, including adjusting the congestion window and perceiving the good path conditions. Then, it can sacrifice a small percentage of network usage caused by redundant packets to improve transmission performance and robustness.

To achieve these goals, EdAR includes two modes: normal transmission and redundant transmission, and utilizes a redundant buffer to make smooth switching between them. The redundant buffer stores packets that have been sent once but have not yet been acknowledged (ACKed) yet. If a new packet is sent on subflows from the send buffer, it will be copied to the redundant buffer. Otherwise, if a packet in the redundant buffer is sent out or ACKed, it will be removed from the redundant buffer. To be noted, since MPTCP needs to keep the data packets before they are ACKed to provide reliable transmission, EdAR does not include extra storage overhead of redundant packets. The redundant buffer is a visual buffer that

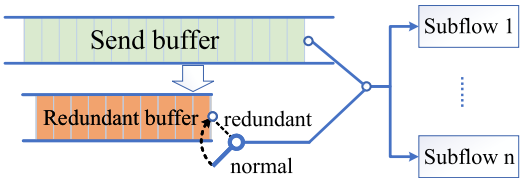


Fig. 4. The packet scheduling strategies in EdAR scheduler.

contains the pointers to the packets. Based on the redundant buffer, the normal and redundant transmissions are defined as:

- **Normal**: In a normal transmission, the redundant buffer is not linked to subflows and all packets will be sent only once. When a subflow is available to send packets, EdAR takes new packets from send buffer and sends packets on the available subflow.
- **Redundant**: In a redundant transmission, the redundant buffer is linked to subflows for redundant packet sending, where a packet can be sent more than once on different subflows before it is ACKed. Once a subflow is available to send a packet, EdAR first checks the redundant buffer. If the first available redundant packet has not been sent on the subflow, EdAR schedules the redundant packet and removes it from the redundant buffer. Otherwise, EdAR schedules a new packet from the send buffer on the subflow.

In addition, if there are more than one subflows available, EdAR always schedules packets on the available subflow with the minimum RTT to fill its congestion window, before scheduling packets on the other subflows.

As shown in Fig. 4, The redundant buffer always stores redundant copies of packets from the send buffer. Once a subflow experiences a dramatic deterioration (e.g., a large increase in packet loss or delay) or interruption, the lost packets and enlarged delay will cause a large number of out-of-order packets or even block the data transmission. A well-trained EdAR scheduler can respond to the changing network conditions and switch from normal to redundant transmission, quickly retransmitting old packets from the redundant buffer to protect transmission performance. With our design, EdAR guarantees a goodput consistent with a single path TCP on the best path if it is always in the redundant transmission, and aggregates bandwidth if it is always in the normal transmission.

B. Design of EdAR Agent

EdAR determines scheduling strategies through a DRL-based approach. It utilize an EdAR agent to perceive the state of each MPTCP connection and learn to choose an appropriate action. The agent interacts with the environment, observes the states, take actions, and receives feedback rewards. With learning from the feedback reward information of the environment, the agent can learn the optimal actions to maximize the long-term cumulative discount reward.

To reduce the computational overhead, EdAR makes decisions in each time interval (TI), where TI is the minimum round trip time (RTT) of all subflows in the MPTCP connection and may change over time. In each time of scheduling, EdAR collects the state of the last TI and determines the scheduling strategy of the next TI. Furthermore, we also test the implementation of making decisions for every packet but not every TI to achieve more flexibility in packet scheduling. However, it does not make an obvious improvement but brings about a high computational overhead. Therefore, we still use the per-TI decision in EdAR. Moreover, since MPTCP sender does not have information of observed states when a connection first starts, EdAR uses redundant transmission at the very beginning of a connection, which can provide robustness and reduce RTT for mice flows. Table I summarizes the notations used in this section.

TABLE I
THE NOTATIONS

Notations	Meanings
$\mathbf{s}_t$	State at time t
a_t	Action at time t
r_t	Reward at time t
d_t^i	Average RTT of subflow i at time t
o_t^i	Average RTO of subflow i at time t
x_t^i	Average throughput of subflow i at time t
w_t^i	Average congestion window of subflow i at time t
l_t^i	Average packet loss rate of subflow i at time t
f_t^i	Average inflight packets of subflow i at time t
b_t	Receive window of connection at time t
I_t	Inflight packet of connection at time t
G_t	Average goodput of connection at time t
Y_t^i	Average receive throughput of subflow i at time t
O_t	Average out-of-order packets at time t
F_t	HoL blocking time at time t
B_t	Receive buffer size at time t
TI_t	Observation and decision time interval at time t
α	Weight of redundant packets overhead in the reward function
β	Weight of HoL blocking in the reward function
$\boldsymbol{\theta}$	Parameters of DRL agent

In the EdAR agent, the state, action, reward, and policy are defined as follows:

1) State: A state $\mathbf{s}_t$ is the information observed by the MPTCP sender. Different from the single path transmission, MPTCP could use a variable number of subflows in a connection and the data transmission is considered in both connection level and subflow level. Therefore, we define the states that take both the connection level states and the different number of subflow level states into account. At each TI, the sender observes the state of the entire connection and each subflow, which is expressed as:

$$\mathbf{s}_t = (\mathbf{M}_t, \mathbf{E}_t^1, \ldots, \mathbf{E}_t^n),$$

where $\mathbf{M}_t$ is the connection state, $\mathbf{E}_t^i$ is the subflow state of subflow i, and n is the number of subflows. In the t-th TI, the connection state is defined as $\mathbf{M}_t = (\mathbf{a}_{t-1}, b_t, I_t)$, where $\mathbf{a}_{t-1}$ is the action at time $t-1$, b_t is the current receive window, I_t is the current inflight packets at connection level. The connection states only include current information, while the subflow states include both current information and historical information. The subflow state $\mathbf{E}_t^i$ of a subflow i is defined as:

$$\mathbf{E}_t^i = \begin{pmatrix} d_t^i, & o_t^i, & x_t^i, & w_t^i, & l_t^i, & f_t^i \\ d_{t-1}^i, & o_{t-1}^i, & x_{t-1}^i, & w_{t-1}^i, & l_{t-1}^i, & f_{t-1}^i \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ d_{t-k}^i, & o_{t-k}^i, & x_{t-k}^i, & w_{t-k}^i, & l_{t-k}^i, & f_{t-k}^i \end{pmatrix},$$

where $d_t^i, o_t^i, x_t^i, w_t^i, l_t^i, f_t^i$ are the average RTT, retransmission timeout (RTO), throughput, congestion window, packet loss rate and inflight packets (at subflow level) of subflow i at time t, respectively. Here, k is a hyperparameter that denotes how much historical information could be included in the subflow state. The system state of EdAR is defined as a combination of connection states and subflow states, which includes $3 + 6 \times n \times (k+1)$ observations.

2) Action: An action $\mathbf{a}_t$ indicates how the agent responds to the observed state. In EdAR, an action is a scheduling decision that determines the current packet scheduling strategy. An action can be described by $\mathbf{a}_t \in \{normal, redundant\}$, which indicates normal transmission and redundant transmission, respectively.

EdAR takes redundant packets to improve the transmission performance when the path conditions are far from optimal. The exact overheads of redundant packet transmission can be totally different according to the ever-changing conditions of network paths and user devices. In most cases, there can be very few redundant packets, costing very low overhead. Typically, when all paths' conditions are good, there will be no redundant packet transmission. When one path's conditions are getting worse (usually occurs in mobile networks), there can be a few redundant packets sent on the "bad" subflow, since the "bad" subflow usually occupies a smaller proportion of the bandwidth of the whole connection. In the worst cases, which rarely occur, all packets will get redundant copies to send and the overhead of redundant transmission is up to half of the overall throughput.

3) Reward: Different from other learning-based schedulers which observe both state and reward at the sender, EdAR utilizes a collector at each MPTCP receiver to observe reward features, considering that MPTCP receiver gives more accurate information and also provides observation of out-of-order packets. At each TI, the receiver observes the average goodput G_t of the entire MPTCP connection, average throughput of each subflow $(Y_t^1, \ldots, Y_t^n)$, average out-of-order packets O_t, receive buffer size B_t and HoL blocking time F_t for calculating the reward function. Among them, B_t is usually remaining stable during the transmission of an MPTCP connection. F_t denotes the time that the receive buffer is fully occupied by out-of-order packets.

In this paper, we consider three key factors of high transmission performance in mobile scenarios: goodput, bandwidth utilization, and out-of-order packets. EdAR combines these three key factors and formulates the reward function as a multiple objective reward function:

$$r_t = V_g - \alpha V_w - \beta V_o,$$

where $\alpha > 0$ and $\beta > 0$ are the weights of V_w and V_o, respectively. And

$$\begin{aligned} V_g &= G_t, \\ V_w &= \frac{\sum_{i=1}^{n} Y_t^i - G_t}{\sum_{i=1}^{n} Y_t^i}, \\ V_o &= \frac{O_t}{B_t} + \frac{F_t}{TI_t}, \end{aligned}$$

where TI_t is the observation and decision time interval of t, which is defined by the minimum RTT of subflows of the last observation: $TI_t = \min_i r_{t-1}^i$.

The goal of EdAR is to improve goodput, achieve high bandwidth utilization, and reduce out-of-order packets in different network environments. In the multiple objective reward function, V_g denotes the goodput achievement by successfully transmitting data, and makes positive contribution to the reward. On the basis of high goodput, a MPTCP scheduler needs to send as less as redundant packet to achieve high bandwidth utilization. V_w denotes the waste of bandwidth by sending redundant packets, which makes negative contribution to the reward. V_g denotes the degree of receive buffer occupancy, which indicates the HoL blocking degree and also makes negative contribution to the reward.

4) Policy: A policy is a rule that the agent uses to decide which action to take. In our framework, we use $\mu_{\boldsymbol{\theta}}(\cdot)$: $\mathcal{S} \rightarrow \mathcal{A}$ to denote the mapping from the observed states of to the actions, where $\boldsymbol{\theta}$ is the set of parameters used in the model of a DRL agent, $\mathcal{S}$ and $\mathcal{A}$ are the state space and action space, respectively. In a learning task, the goal is to maximize the expected cumulative discount rewards: $\max_{\boldsymbol{\theta}}\{\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]\}$, where $\mathbb{E}[\cdot]$ is the expectation function, $\gamma \in (0, 1)$ is a factor in discounting future rewards.

EdAR is based on Deep Q-learning (DQN) [47], which utilizes a deep network to represent the value function, and continuously updates the network until convergence. It trains a neural network (also known as Q-network) to output the Q-value function $Q(\mathbf{s}, \mathbf{a}|\boldsymbol{\theta})$, and utilizes greedy policy for generating action: $\mu_{\boldsymbol{\theta}}(\mathbf{s}) = \arg\max_{\mathbf{a}}\{Q(\mathbf{s}, \mathbf{a}|\boldsymbol{\theta})\}$, where $Q(\mathbf{s}, \mathbf{a}|\boldsymbol{\theta})$ is the output Q-value.

C. Q-Network of EdAR

The learning task of EdAR in mobile scenarios faces different challenges. First, due to the multiple paths used in MPTCP, different MPTCP flows can have a different number of subflows, that make the different dimensions of observed states. Second, to deal with the mobile networks, an well-designed scheduler needs ability to perceive the changing network conditions and predict future conditions for decision making. Therefore, the Q-network of EdAR should be able to deal with the different input sizes of subflows, and also jointly take the connection state into consideration. To deal with it, we design a special Q-network as shown in Fig. 5.

The Q-network of EdAR consists of two parts: a representation network and a core network. The representation network is used to extract representative features from subflow states, which can deal with the changing network conditions and different number of subflows. The input of representation network is $(\mathbf{E}_t^1, \ldots, \mathbf{E}_t^n)$ and output is $\mathbf{h}_{\boldsymbol{\theta}}(\mathbf{E}_t^1, \ldots, \mathbf{E}_t^n)$. The core network is a fully connected convolutional neural network (CNN), which is used to represent the complex relationship of different input features. The input of the core network can be written as $\widetilde{\mathbf{s}}_t = \{\mathbf{M}_t, \mathbf{h}_{\boldsymbol{\theta}}(\mathbf{E}_t^1, \ldots, \mathbf{E}_t^n)\}$, which consists of the connection states and the representation output of subflow states. The output of core network is $Q(\mathbf{s}_t, \mathbf{a}_t|\boldsymbol{\theta})$, which is the approximation of long-term accumulative discount reward.

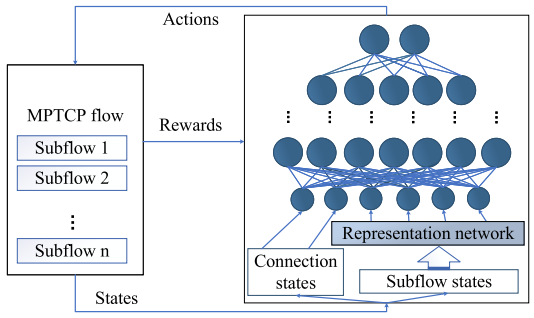


Fig. 5. Q-network of EdAR.

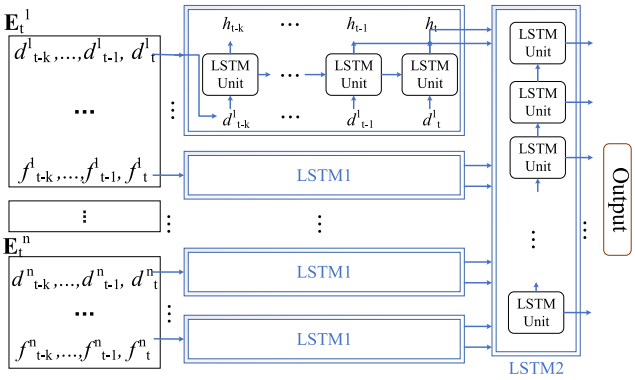


Fig. 6. Representation network of EdAR.

Fig. 6 shows the structure of the representation network, which consists of several Long Short Term Memory (LSTM) networks [48]. First, a series of LSTM networks extract subflow features from historical states, where each LSTM network has the same structure and can be called LSTM1. The input of each LSTM1 is a historical series of a single subflow state, such as $\{d^1_{t-k}, d^1_{t-k+1}, \ldots, d^1_t\}$, and the output is the last two hidden states of the LSTM network. In the constructed representation network, the last two hidden states of LSTM1 represent the prediction of the average state and trends of each time series of data, respectively. After that, an LSTM with a different structure (also called LSTM2) is used to deal with the input of the different number of MPTCP subflows. Results from each LSTM1 are then aggregated as the input of LSTM2. The output of LSTM2 is the last 24 hidden states.

Both LSTM1 and LSTM2 in the representation network are represented as single-layer LSTM. Each LSTM1 has 10 LSTM units, and LSTM1 has 60 LSTM units. The core network has two hidden layers that contain 128 and 32 units, respectively. Each layer is activated by a Rectified Linear (ReLU) function and the output layer is not activated.

V. Asynchronous Learning Approaches

In this section, we propose the learning approach for EdAR considering the following challenges:

- **Cold start**: A DRL agent usually has a cold start time with poor performance until convergence, where it may take actions that are far from the optimal.

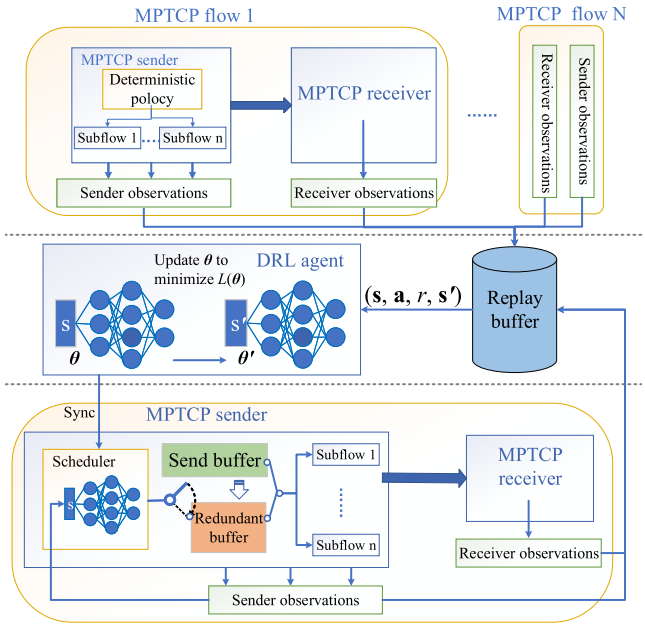


Fig. 7. EdAR framework.

- **Asynchronous**: For MPTCP transmissions, states and rewards need to be collected separately on the sender and receiver.
- **Environment diversity**: Different MPTCP servers face different environments and the network environment is constantly changing.

To address these challenges, as shown in Fig. 7, EdAR integrates both offline and online learning approaches. The offline learning approach is used to provide a pre-trained model which adapts to most environments with baseline performance. The online learning approach is conducted individually on each MPTCP server using asynchronous learning methods, reducing computational complexity while adapting to each server's target network environments.

A. Loss Function and Learning Process

EdAR utilizes the gradient descent to train the network parameters and minimize the loss function. There are two Q-networks (target Q-network and main Q-network) in the model, which have the same structure. The loss function is defined as:

$$\begin{aligned} L(\boldsymbol{\theta}) &= \mathbb{E}\left[\left(y_t - Q(\mathbf{s}_t, \mathbf{a}_t; \boldsymbol{\theta})\right)^2\right] \\ &= \mathbb{E}\Bigg[\Big(r_t + \gamma \max_{\mathbf{a}_{t+1}}\{Q(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}; \boldsymbol{\theta}')\} \\ &\quad - Q(\mathbf{s}_t, \mathbf{a}_t; \boldsymbol{\theta})\Big)^2\Bigg], \end{aligned}$$

where $\boldsymbol{\theta}'$ denotes the parameters of the target Q-network, $Q(\mathbf{s}, \mathbf{a}; \boldsymbol{\theta}')$ denotes the output of the target Q-network, $Q(\mathbf{s}, \mathbf{a}; \boldsymbol{\theta})$ denotes the output of the main Q-network, and $y_t = r_t + \gamma \cdot \max_{\mathbf{a}_{t+1}}\{Q(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}; \boldsymbol{\theta}')\}$ denotes the target Q-value. Using two networks reduces the correlation between the current Q-value and the target Q-value, and improves the

stability. The weight of the target Q-network updates slowly to track the main Q-network:

$$\boldsymbol{\theta}' = (1-\tau)\boldsymbol{\theta}' + \tau\boldsymbol{\theta},$$

where $\tau < 1$ is the updating rate. This setting is used to make slow change of the constraint target value, which significantly improves the stability of the learning approach.

B. Offline Learning

Since all parameters of the Q-network are randomly initialized, an agent cannot entirely rely on network-derived actions early in learning. As shown in Fig. 7, EdAR adopts an offline learning approach that uses data collected from all MPTCP flows in the network to expand the diversities of data and pre-trains a model for baseline performance. In the offline learning approach, MPTCP senders do not use the agent to generate actions. Instead, two determined policies, $\mu(\mathbf{s}) = normal$ and $\mu(\mathbf{s}) = redundant$ are used, which represent always using normal transmission and redundant transmission, respectively. This simplifies the process of collecting data for offline learning, where the MPTCP scheduler does not need to add additional functionality to generate action in the offline data collection. In addition, the data collector runs on both sender and receiver of each MPTCP flow to collect data. The observation collected by the data collector that includes transitions in the form of $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$ is stored in the replay buffer for offline learning.

The offline learning algorithm is given in Algorithm 1. It runs only once and uses collected data from all MPTCP flows to train a pre-trained model. After that, MPTCP flows can use the pre-trained model for achieving baseline performance.

Algorithm 1 Offline Learning and Data Collection

```
/* Data collection */
for each MPTCP flow do
    Randomly initialize μ(s) = normal or μ(s) = redundant;
    for t ∈ [1, T] do
        Sender observes s_t and execute a_t = μ(s_t);
        Sender observes s_{t+1};
        Receiver observes r_t;
    Send all transitions {(s_t, a_t, r_t, s_{t+1})} to the replay buffer;
/* Offline learning */
Input: Collected transitions
Output: Q-network parameters Θ
Initialize the replay buffer with transitions of (s_t, a_t, r_t, s_{t+1});
Randomly initialize the main Q-network with normalized parameters θ;
Initialize the target Q-network with the parameters as the main Q-network θ' ← θ;
for iteration ∈ [1, M] do
    Sample a random minibatch of H transitions of (s_t, a_t, r_t, s_{t+1}) from the replay buffer;
    if the MPTCP flow ends after t then
        y_t = r_t ;
    else
        y_t = r_t + γ max_{a_{i+1}} {Q(s_{t+1}, a_{t+1}|θ')};
    Update the weight θ by minimizing the loss: L = (1/H) Σ(y_t − Q(s_t, a_t|θ))^2;
    Update the target Q-network: θ' ← τθ + (1 − τ)θ'
return Θ = θ;
```

C. Asynchronous Online Learning and Scheduling

EdAR utilizes an asynchronous online learning approach to support parallel data collection on sender and receiver. At the same time, it decouples data collection and model learning so that real-time scheduling and learning can be performed in parallel. After synchronizing the pre-trained model, each MPTCP server creates its own agent and retrains a customized model online for adopting different environments. An agent serves all the MPTCP flows on the server, constantly using the latest experience collected from the flows (at both the sender and receiver) to train the Q-network, and iteratively synchronizes its Q-network parameters with the scheduler. When the network environment changes, EdAR can learn from new data and adapt to the new environment as soon as possible.

The online learning and scheduling algorithm is given in Algorithm 2. EdAR runs a complete online scheduling procedure and a complete online learning episode for each MPTCP flow,respectively. During the online scheduling procedure, EdAR observes a set of original network signals and subscribes as a state input to the agent to output an action. The reward is then observed at the receiver. After an MPTCP flow is completed, the collected states, actions, and rewards are used to update the replay buffer for learning and improving the agent. Due to the capacity limitation of the replay buffer, the old transitions will be deleted if the replay buffer is full. Only the recent collected transitions will be used for online learning. According to the real network trajectory collected, each server can train its customized agent. Note that inexperienced agents need to make full use of the random transfer samples to explore and gain the necessary good and bad experiences and ultimately learn a good (hopefully best) policy. EdAR uses a ϵ-$greedy$ policy to generate action, which combines "exploration" and "exploitation" with probability $\epsilon < 1$. It randomly chooses an action from action space with the probability ϵ, and chooses the action $\mathbf{a} = \arg\max_{\mathbf{a}}\{Q(\mathbf{s}, \mathbf{a}|\boldsymbol{\theta})\}$ with the probability $(1-\epsilon)$. Randomly choosing an action guarantees the agent go through all the actions in the action space, therefore providing extra information for the agent to train the Q-network. The ϵ-$greedy$ policy is only used to generate actions in the online learning approach. After the online learning ends, that is, the learning loss converges to a lower value and the server does not need to collect more data to train the model. Then the server will close the online learning approach and use a greedy policy where $\mu_{\boldsymbol{\theta}}(\mathbf{s}) = \arg\max_{\mathbf{a}}\{Q(\mathbf{s}, \mathbf{a}|\boldsymbol{\theta})\}$ to generate action.

Algorithm 2 Online Learning and Scheduling

Initialize an empty replay buffer;
Initialize the pre-trained main Q-network parameters $\boldsymbol{\theta} \leftarrow \boldsymbol{\Theta}$;
Initialize the target Q-network with the parameters as the main Q-network $\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta}$;
for *each MPTCP flow* **do**
  for *each time* t **do**
    if $random() < \epsilon$ **then**
      randomly choose $\mathbf{a}_t = normal$ or $\mathbf{a}_t = redundant$;
    else
      $\mathbf{a}_t = \arg\max_{\mathbf{a}_t}\{Q(\mathbf{s}_t, \mathbf{a}_t|\boldsymbol{\theta})\}$;
    Sender executes $\mathbf{a}_t$ and observes $\mathbf{s}_{t+1}$;
    Receiver observes r_t;
  Send all transitions of $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$ to the replay buffer;
  for $iteration \in [1, M]$ **do**
    Sample a random minibatch of H transitions of $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$ from the replay buffer;
    if *the MPTCP flow ends after* t **then**
      $y_t = r_t$;
    else
      $y_t = r_t + \gamma \max_{\mathbf{a}_{t+1}}\{Q(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}|\boldsymbol{\theta}')\}$;
    Update the weight $\boldsymbol{\theta}$ by minimizing the loss: $L = \frac{1}{H}\sum(y_t - Q(\mathbf{s}_t, \mathbf{a}_t|\boldsymbol{\theta}))^2$;
    Update the target Q-network: $\boldsymbol{\theta}' \leftarrow \tau\boldsymbol{\theta} + (1-\tau)\boldsymbol{\theta}'$
  Synchronize $\boldsymbol{\theta}$ to the scheduler;

An agent needs to be retrained when the network environment changes, such as changing from low-bandwidth networks to high-bandwidth networks or changing from low-latency networks to high-latency networks. Since the old experience will bring a deviation in the new environment, it needs to collect transition samples and update the agent to gain enough experience for making the right decisions. For EdAR, if the network changes, it only needs to restart an online learning approach and synchronize the model at the current server.

We analyze the computation overhead of EdAR when using it to enhance the transmission performance in real-time. Since the learning approaches are decoupled with online scheduling, they will not increase the computational overhead during transmission. The following shows the computation overhead of online decisions, which depend on the network structure. The Q-network of EdAR consists of $L \times n$ LSTM1, one LSTM2, and a core network, where L is the number of subflow statements, and n is the number of subflows. The overall computation complexity of LSTM is linear to its units. Then, the computation complexity $L \times n$ LSTM1 and an LSTM2 is $O(L \times n \times K_1 + K_2)$, where K_1, K_2 are the number of units of LSTM1 and LSTM2, respectively. The computation complexity of the core network depends on the input size and the units of each layer. Let C_1, C_2, and C_3 denote the input size, units of the first layer, and units of the second layer, respectively, and the computation complexity of the core network is $O(C_1 \times C_2 + C_2 \times C_3 + C_3)$. The overall computation complexity of one time online decision is $O(L \times n \times K_1 + K_2 + C_1 \times C_2 + C_2 \times C_3 + C_3)$, where n depends on the actual using subflows, and L, K_1, K_2, C_1, C_2, C_3 depend on the network structure. For using EdAR in mobile networks, the number of subflows is limited (usually ≤ 3) and the network structure of Q-network is small, so the computation overhead of making real-time decisions is acceptable.

VI. Performance Evaluation

In this section, we evaluate EdAR in the NS-3 simulator [49] and evaluate its performance in different network scenarios. The parameter settings in the implementation of EdAR are $k = 5$, $\alpha = 1.0$, and $\beta = 0.5$, respectively. The Q-network is trained by Adam Optimizer, with a learning rate of 0.0001, $\tau = 0.01$, and the discount factor $\gamma = 0.6$. We compare EdAR with the existing mechanisms with the model-based approaches (minRTT [37], BLEST [50]) and the learning-based approaches (Peekaboo [36]) in the evaluations, where we implement Peekaboo in NS-3 using the same parameter setting in [36].

A. Simulation Settings

For today's commonly used mobile devices, such as cell phones and tablets, there are usually two different wireless interfaces, where an MPTCP connection usually establishes two subflows over different paths. We let each MPTCP flow contain two subflows in the simulation, which uses the same topology as Fig. 1. The path characteristics are chosen from Table II, where each characteristic is randomly chosen from the range given in the table. We configure five different settings with different ranges for the path characteristics, where each path is characterized by a 5-tuple including its bandwidth, delay, random packet loss rate, and whether handoff happens. If handoff is enabled, the last hop link of a path randomly suffers a mobile link and causes handoff every 40s (starts at time t_1 and ends at time t_2). Handoff always gives rise to a decrease of link quality. The link between the MPTCP receiver and router is a mobile link. The user starts moving away from the router at t_1 until the link is unavailable, and moves back at t_2. After t_2, the link return to the initial state.

EdAR utilizes both offline learning and online learning approaches. We use EdAR-p and EdAR-o to denote the EdAR scheduler with offline learning and online learning approaches, respectively. Only data collected from settings i, ii, and iii are used for offline learning, where we collect 100 different trajectories in each setting of transmitting 10-100 MB (with a uniform distribution) data. The performance evaluations in settings i, ii, and iii use the same ranges of the path characteristics as offline learning data collection, but with different random realizations of the characteristics. Each episode of online training contains a complete trajectory of an MPTCP flow. After a flow completes, both the sender and receiver send the collected data to the server. The replay buffer is then updated with the new trajectory and removes the outdated

TABLE II
PATH CHARACTERISTICS SETTINGS FOR DIFFERENT NETWORK ENVIRONMENTS

	setting i		setting ii		setting iii		setting iv		setting v	
	path 1	path 2	path 1	path 2	path 1	path 2	path 1	path 2	path 1	path 2
bandwidth (Mbps)	5-10	5-10	5-10	5-10	5-10	1-5	20-30	5-10	0.1-0.5	0.1-0.5
delay (ms)	20-30	20-30	20-50	50-100	20-30	50-70	5-10	10-20	50-100	100-200
loss rate (%)	0.1-0.5	0.1-0.5	0.1-0.5	0.1-1.0	0.1-0.5	0.1-0.5	0.1-0.5	0.5-1.0	1.0-2.0	1.0-2.0
handoff	non	enabled	non	enabled	non	enabled	non	enabled	non	enabled
receive buffer (KB)	128		128		128		256		64	

TABLE III
PATH CHARACTERISTICS

	bandwidth	delay	loss rate	handoff
path 1	10 Mbps	25 ms	0.1 %	non
path 2	10 Mbps	50 ms	0.1 %	enabled

data. Then, the agent uses the data in the replay buffer to train the Q-network, and synchronizes the trained Q-network parameters with the scheduler. Usually, the online training converges after several or tens of training episodes (depending on how much the network environment deviates from the offline training data). Moreover, the offline and online learning approaches are decoupled with online scheduling, which will not increase the computational overhead of online scheduling. When applying the model for real-time scheduling decision, its computation time of generating an action from a given state is less than 10^{-1} ms. The computation is conducted on Intel Core i7-12700K CPU, 16GB Random Access Memory (RAM). The computational overhead is acceptable since the online scheduling makes one time decision for every TI (usually tens of milliseconds), which only occupy a small part of time during the data transmission.

B. Simulation Results

We first show the comparison of different schedulers where handoff happens during the transmission. The specific path characteristics are shown in Table III, where path 2 enables handoffs and suffers handoff at 20-40s and 60-80s. The path characteristics are in the range of setting ii, which means that an EdAR pre-trained model could provide a good performance. We illustrate the results of continuing data transmission during 0-100s. During 40-60s and 60-80s, the goodput of minRTT, BLEST, and Peekaboo significantly decreases, even though path 1 still has the ability to provide 10 Mbps throughput. Among them, minRTT has the lowest goodput, since it always sends data on all paths and the packet loss on path 2 will block the transmission. BLEST and Peekaboo are better than minRTT, because they can decide to pause a subflow with a high RTT to reduce HoL blocking and enhance performance. However, if packet loss has already occurred on a "handoff" subflow and block the data transmission, they still need to wait for the packet retransmission and thus cause a goodput degradation.

Fig. 8 shows the real-time goodput during the transmission. The goodput of minRTT, BLEST, and Peekaboo decreases to less than the available goodput on the best path during

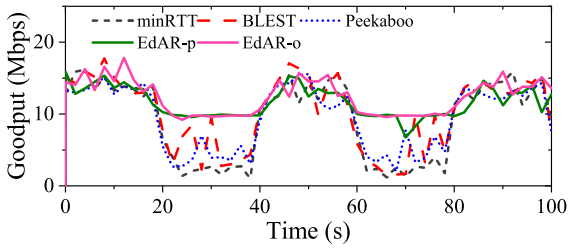


Fig. 8. Real-time goodput of different schedulers under a handoff situation.

20-40 s and 60-80 s. Compared with them, EdAR can utilize redundant packets to provide high robustness, while the packet loss on a subflow can be compensated by redundant packets send on the other subflows where the path condition is better. Moreover, by learning from old data, EdAR is able to switch its scheduling strategies according to real-time states and provide high goodput under different situations. In this scenario, the performance of EdAR-p and EdAR-o is almost identical. Both are able to maintain the highest goodhput during handoff states while aggregating bandwidth in stable states. This is because the pre-trained dataset contains the data with similar parameters to this scenario, so the pre-trained model is able to perform well.

Fig. 9 shows the statistics of goodput and out-of-order packets during the transmission. Fig. 9(a) shows the average goodput during both stable and handoff situations. "Stable" denotes the statistics during 0-20 s, 40-60 s, and 80-100 s, where handoff does not happen on both the two paths. "Handoff" denotes the statistics during 20-40 s, and 60-80 s, where handoff happens on path 2. In stable situations, all the schedulers can aggregate bandwidth and provide a high goodput. When one path is suffering handoffs, the overall goodput decreases. Among different schedulers, the goodput of minRTT is the lowest, which decreases to 2.4 Mbps. The goodput of BLEST and Pekaboo is better than that of minRTT but is still as good as expected. Compared with them, EdAR-p and EdAR-o outperform the other comparison schemes during handoff situations. Among them, EdAR-o is more stable than EdAR-p, since it retrains the model using the recent data, therefore it is more suitable to current situations. Fig. 9(b) shows the distribution of out-of-order packets. Compared with others, EdAR-p and EdAR-o reduce the out-of-order packets, which gives the credit to the redundant packet sending during handoffs. However, the out-of-order packet of EdAR-o does not always outperform EdAR-p. On the one hand, it is caused by the randomness of the path states. On the other hand, out-of-order packets are not the only factor in the reward function.

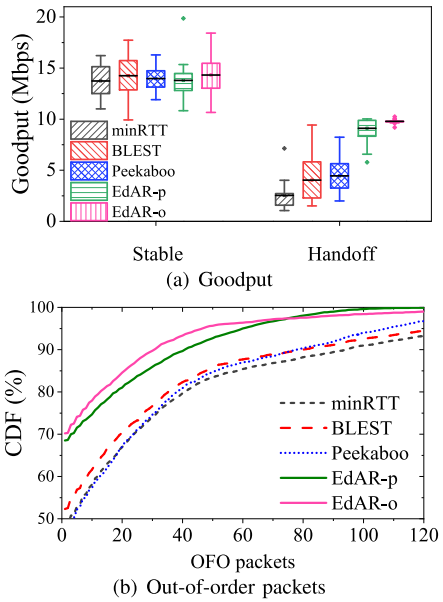


Fig. 9. Comparison of goodput and out-of-order packets of different schedulers under a handoff situation.

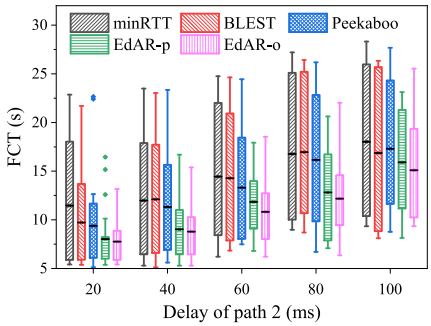


Fig. 10. Comparison of flow completion time with different path delays.

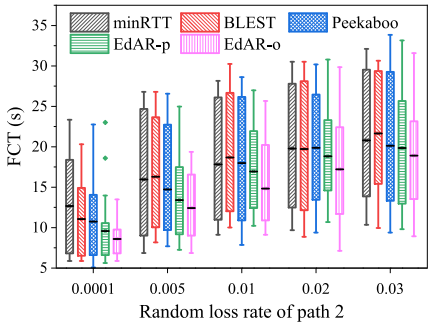


Fig. 11. Comparison of flow completion time with different random loss rates.

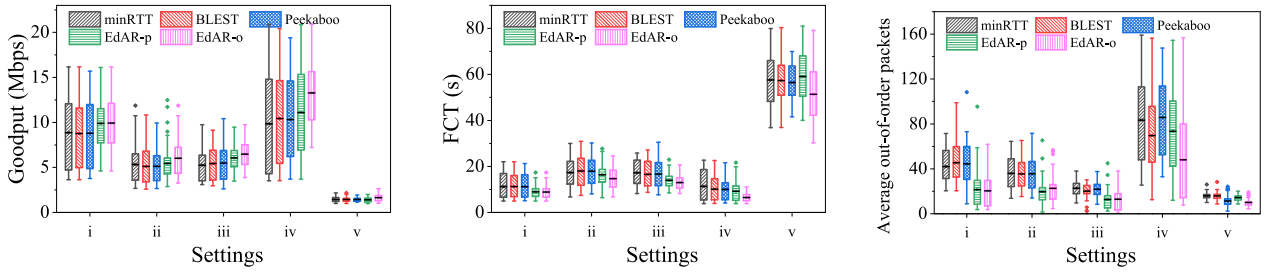
The overall reward function is a combination of multiple metrics. Because the setting is not very different from the parameter settings in the offline training set, the performance gap between EdAR-o and EdAR-p is not obvious. But overall, EdAR-o is slightly better than EdAR-p.

We then change the delay and loss rate of path 2 and show the performance of different schedulers in Fig. 10 and Fig. 11. The bandwidth, delay and loss of path 1 are set to 10 Mbps, 25 ms, and 0.1%, respectively. In Fig. 10, the bandwidth and RTT of path 2 are keeping as 10 Mbps and 25 ms, respectively, and the random loss rate of path 2 is varying from 0.01% to 3.0%. In Fig. 11, the bandwidth and random loss rate of path 2 are keeping as 10 Mbps and 0.1% respectively, and the delay of path 2 is varying from 20 to 100 ms. We illustrate the flow completion time (FCT) of downloading a 10MB file for 50 times at each parameter setting to show the simulation results of statistics.

As shown in Fig. 10, with the delay of path 2 increases, the overall FCT of MPTCP gradually increases, which means the overall performance of MPTCP degrades. On the one hand, this is partly due to the poor transmission performance of path 2. On the other hand, the increase of the asymmetry of the two paths leads to the increase of out-of-order packets and thus affects the overall transmission. Among different schedulers, minRTT and BLEST usually perform the highest FCT. Since there are unpredictable handoffs during the data transmission, minRTT and BLEST also have the highest fluctuation in FCT. Peekaboo is better than minRTT and BLEST. However, it also can not perform well in high mobile scenarios. EdAR-p and EdAR-o reduce the FCT a lot under different parameter settings. Since redundant packets are utilized to improve robustness during handoff time, EdAR-p and EdAR-o also have the lowest fluctuation in FCT.

As shown in Fig. 11, as the random loss rate of path 2 increases, the overall FCT of MPTCP also gradually increases. This is not only because the packet loss on path 2 causes a degradation of the single path performance, but also because the packet loss on a single path affects the transmission on the entire MPTCP connection. Compared with minRTT, BLEST, and Peekaboo, EdAR-p and EdAR-o provide the shortest FCT. The advantage of EdAR-p and EdAR-o is more obvious in the case of a low random loss rate. This is because a low random loss rate means that the path state is stable, so the scheduler can make more accurate judgments. Moreover, in the situation where the random loss rate of path 2 is more than 1%, EdAR-p has almost the same performance as Peekaboo and minRTT, which only shows a small performance improvement. This is due to the large gap between such a scenario and the data in the previous training set. Therefore the pre-trained model does not bring much improvement. Moreover, after online training, EdAR-o further improves the transmission performance substantially.

Moreover, we also compare the performance of different schedulers under different path parameter settings in Table II. For different path parameter settings, they take different ranges of bandwidth, delay, and packet loss rate, as well as different receive buffer size. Among the different path parameter settings, setting i is set as the asymmetric network, which has the same ranges of bandwidth, delay, and packet loss rate. Settings ii and iii are set to asymmetric networks with different ranges of path delay, bandwidth, and loss rate. Settings iv and v are far away from settings i, ii, and iii of the ranges of path delay, bandwidth, and loss rate. Among them, settings iv reflects to high bandwidth and low delay networks, settings v reflects to low bandwidth and high delay networks. In addition, the receive buffer size of settings iv and v is different from settings i, ii, and iii. Only the data collected from setting i, ii, and iii is used in the offline learning approach, settings iv and v are



(a) Average goodput (b) Average FCT (c) Out-of-order packets

Fig. 12. Comparison of goodput, FCT, and out-of-order packets under different network settings.

considered as new environments that do not have collected data for offline learning. Therefore, EdAR-p that utilizes the pre-trained model is more suitable in settings i, ii, and iii.

Fig. 12 shows the comparison of goodput, FCT, and out-of-order packets under different network settings. In settings i, ii, and iii, EdAR-p and EdAR-o outperform minRTT, BLEST, and Peekaboo in average goodput, FCT, and out-of-order packets, as well as making the lowest fluctuation in their performance. Since the pre-trained model has already got experience from historical data, EdAR-p is well trained for these environments and could provide good performance. EdAR-o only makes a slight improvement based on EdAR-p. Among them, in settings i, the total goodput is the highest and EdAR-p brings the largest improvement on the overall goodput. Because setting i is a symmetric network scenario with low loss rates. In such a more stable network environment, the pre-trained model can make more accurate decisions. Compared with it, EdAR-o brings a more improvement in setting ii and iii.

On the contrary, since the pre-trained model does not have enough experience of settings iv and v, EdAR-p does not make a significant improvement. Among them, setting v is significantly different from the offline learning data set, which makes EdAR-p underperform other schedulers. After collecting new data and going through an online learning approach, EdAR-o makes an obvious improvement of transmission efficiency and robustness compared to EdAR-p in settings iv and v, providing the highest goodput, lowest FCT, and lowest out-of-order packet among all the schedulers.

VII. Conclusion

In this paper, we introduced the design of EdAR, an experience-driven multipath scheduler that presents a novel trainable model with flexible redundant packet scheduling to deal with the handoff issue in wireless mobile networks. A well-trained EdAR scheduler has the ability to respond quickly to the changing network conditions, and to take accurate scheduling decisions in different network environments. The experimental results also show the effectiveness of EdAR, whose robustness is much better than that of state-of-the-art multipath schedulers, especially in mobile scenarios.

References

[1] H. Sinky, B. Hamdaoui, and M. Guizani, "Seamless handoffs in wireless HetNets: Transport-layer challenges and multi-path TCP solutions with cross-layer awareness," *IEEE Netw.*, vol. 33, no. 2, pp. 195–201, Mar. 2019.

[2] F. Le and E. M. Nahum, "Experiences implementing live VM migration over the WAN with multi-path TCP," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr./May 2019, pp. 1090–1098.

[3] L. Li et al., "A measurement study on multi-path TCP with multiple cellular carriers on high speed rails," in *Proc. Conf. ACM Special Interest Group Data Commun. (SIGCOMM)*, 2018, pp. 161–175.

[4] B. Liu, J. Liu, and N. Kato, "Optimal beamformer design for millimeter wave dual-functional radar-communication based V2X systems," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 10, pp. 2980–2993, Oct. 2022.

[5] G. Chen et al., "Fast and cautious: Leveraging multi-path diversity for transport loss recovery in data centers," in *Proc. USENIX Annu. Tech. Conf. (ATC)*, 2016, pp. 1–15.

[6] Z. Shi and J. Liu, "Sparse code multiple access assisted resource allocation for 5G V2X communications," *IEEE Trans. Commun.*, vol. 70, no. 10, pp. 6661–6677, Oct. 2022.

[7] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving datacenter performance and robustness with multipath TCP," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 266–277, 2011.

[8] A. Ford, C. Raiciu, M. J. Handley, O. Bonaventure, and C. Paasch, *TCP Extensions for Multipath Operation With Multiple Addresses*, document RFC 8684, 2020. Accessed: May 2022. [Online]. Available: https://rfc-editor.org/rfc/rfc8684.txt

[9] C. F. Silva, S. Ferlin, O. Alay, A. Brunstrom, and B. Y. L. Kimura, "IoT traffic offloading with MultiPath TCP," *IEEE Commun. Mag.*, vol. 59, no. 4, pp. 51–57, Apr. 2021.

[10] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath TCP," in *Proc. 8th USENIX Conf. Netw. Syst. Design Implement. (NSDI)*, 2011, pp. 1–14.

[11] J. Han, K. Xue, W. Wei, Y. Xing, J. Liu, and P. Hong, "Transparent multipath: Using double MPTCP proxies to enhance transport performance for traditional TCP," *IEEE Netw.*, vol. 35, no. 5, pp. 181–187, Sep. 2021.

[12] M. Li, A. Lukyanenko, Z. Ou, A. Ylä-Jääski, S. Tarkoma, M. Coudron, and S. Secci, "Multipath transmission for the internet: A survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 4, pp. 2887–2925, 4th Quart., 2016.

[13] S. D. Sathyanarayana, J. Lee, J. Lee, D. Grunwald, and S. Ha, "Exploiting client inference in multipath TCP over multiple cellular networks," *IEEE Commun. Mag.*, vol. 59, no. 4, pp. 58–64, Apr. 2021.

[14] J. Han et al., "Leveraging coupled BBR and adaptive packet scheduling to boost MPTCP," *IEEE Trans. Wireless Commun.*, vol. 20, no. 11, pp. 7555–7567, Nov. 2021.

[15] Y. Zhang, J. Tourrilhes, Z.-L. Zhang, and P. Sharma, "Improving SD-WAN resilience: From vertical handoff to WAN-aware MPTCP," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 1, pp. 347–361, Mar. 2021.

[16] H. Shi et al., "STMS: Improving MPTCP throughput under heterogeneous networks," in *Proc. 2018 USENIX Annu. Tech. Conf. (ATC)*, 2018, pp. 719–730.

[17] S. R. Pokhrel, M. Panda, and H. L. Vu, "Analytical modeling of multipath TCP over last-mile wireless," *IEEE/ACM Trans. Netw.*, vol. 25, no. 3, pp. 1876–1891, Jun. 2017.

[18] S. Ferlin, S. Kucera, H. Claussen, and O. Alay, "MPTCP meets FEC: Supporting latency-sensitive applications over heterogeneous networks," *IEEE/ACM Trans. Netw.*, vol. 26, no. 5, pp. 2005–2018, Oct. 2018.

[19] H. Lee, J. Flinn, and B. Tonshal, "RAVEN: Improving interactive latency for the connected car," in *Proc. 24th Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, 2018, pp. 557–572.

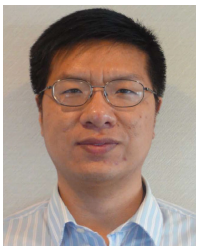
[20] W. Wei, K. Xue, J. Han, D. S. L. Wei, and P. Hong, "Shared bottleneck-based congestion control and packet scheduling for multipath TCP," *IEEE/ACM Trans. Netw.*, vol. 28, no. 2, pp. 653–666, Apr. 2020.
[21] Y. Xing et al., "A low-latency MPTCP scheduler for live video streaming in mobile networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 11, pp. 7230–7242, 2021.
[22] A. Nikravesh, Y. Guo, F. Qian, Z. M. Mao, and S. Sen, "An in-depth understanding of multipath TCP on mobile devices: Measurement and system design," in *Proc. 22nd Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, 2016, pp. 189–201.
[23] J. Han, K. Xue, Y. Xing, P. Hong, and D. S. L. Wei, "Measurement and redesign of BBR-based MPTCP," in *Proc. ACM SIGCOMM Conf. Posters Demos*, Aug. 2019, pp. 75–77.
[24] B. Jin, S. Kim, D. Yun, H. Lee, W. Kim, and Y. Yi, "Aggregating LTE and Wi-Fi: Toward intra-cell fairness and high TCP performance," *IEEE Trans. Wireless Commun. (TWC)*, vol. 16, no. 10, pp. 6295–6308, 2017.
[25] Q. De Coninck and O. Bonaventure, "Multiflow QUIC: A generic multipath transport protocol," *IEEE Commun. Mag.*, vol. 59, no. 5, pp. 108–113, May 2021.
[26] M. Morawski and P. Ignaciuk, "A green multipath TCP framework for industrial Internet of Things applications," *Comput. Netw.*, vol. 187, Mar. 2021, Art. no. 107831.
[27] Y. Xing, J. Han, K. Xue, J. Liu, M. Pan, and P. Hong, "MPTCP meets big data: Customizing transmission strategy for various data flows," *IEEE Netw.*, vol. 34, no. 4, pp. 35–41, Jul. 2020.
[28] J. Zhao, J. Liu, H. Wang, C. Xu, W. Gong, and C. Xu, "Measurement, analysis, and enhancement of multipath TCP energy efficiency for data-centers," *IEEE/ACM Trans. Netw.*, vol. 28, no. 1, pp. 57–70, Feb. 2020.
[29] C. Raiciu, M. J. Handley, and D. Wischik, *Coupled Congestion Control for Multipath Transport Protocols*, document RFC 6356, 2011. Accessed: May 2022. [Online]. Available: https://rfc-editor.org/rfc/rfc6356.txt
[30] H. Sinky, B. Hamdaoui, and M. Guizani, "Proactive multipath TCP for seamless handoff in heterogeneous wireless access networks," *IEEE Trans. Wireless Commun.*, vol. 15, no. 7, pp. 4754–4764, Jul. 2016.
[31] C. Xu, J. Zhao, and G.-M. Muntean, "Congestion control design for multipath transport protocols: A survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 4, pp. 2948–2969, 4th Quart., 2016.
[32] S. R. Pokhrel and A. Walid, "Learning to harness bandwidth with multipath congestion control and scheduling," *IEEE Trans. Mobile Comput.*, vol. 22, no. 2, pp. 996–1009, Feb. 2023.
[33] Y.-S. Lim, E. M. Nahum, D. Towsley, and R. J. Gibbens, "ECF: An MPTCP path scheduler to manage heterogeneous paths," in *Proc. 13th Int. Conf. Emerg. Netw. Exp. Technol. (CoNEXT)*, 2017, pp. 147–159.
[34] P. Hurtig, K.-J. Grinnemo, A. Brunstrom, S. Ferlin, O. Alay, and N. Kuhn, "Low-latency scheduling in MPTCP," *IEEE/ACM Trans. Netw.*, vol. 27, no. 1, pp. 302–315, Feb. 2019.
[35] H. Zhang, W. Li, S. Gao, X. Wang, and B. Ye, "ReLeS: A neural adaptive multipath scheduler based on deep reinforcement learning," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2019, pp. 1648–1656.
[36] H. Wu, O. Alay, A. Brunstrom, S. Ferlin, and G. Caso, "Peekaboo: Learning-based multipath scheduling for dynamic heterogeneous environments," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 10, pp. 2295–2310, Oct. 2020.
[37] O. Bonaventure, C. Paasch, and G. Detal, *Use Cases and Operational Experience With Multipath TCP*, document RFC 8041, 2017. Accessed: May 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc8041
[38] Q. De Coninck and O. Bonaventure, "Multipath QUIC: Design and evaluation," in *Proc. 13th Int. Conf. Emerg. Netw. Exp. Technol. (CoNEXT)*, 2017, pp. 160–166.
[39] W. Li, H. Zhang, S. Gao, C. Xue, X. Wang, and S. Lu, "SmartCC: A reinforcement learning approach for multipath TCP congestion control in heterogeneous networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 11, pp. 2621–2633, Nov. 2019.
[40] Z. Shi, J. Liu, S. Zhang, and N. Kato, "Multi-agent deep reinforcement learning for massive access in 5G and beyond ultra-dense NOMA system," *IEEE Trans. Wireless Commun.*, vol. 21, no. 5, pp. 3057–3070, May 2022.
[41] B. Mao, F. Tang, Z. M. Fadlullah, and N. Kato, "An intelligent route computation approach based on real-time deep learning strategy for software defined communication systems," *IEEE Trans. Emerg. Topics Comput.*, vol. 9, no. 3, pp. 1554–1565, Jul. 2021.
[42] B. Mao et al., "A novel non-supervised deep-learning-based network traffic control method for software defined wireless networks," *IEEE Wireless Commun.*, vol. 25, no. 4, pp. 74–81, Aug. 2018.
[43] A. Sacco, M. Flocco, and P. Torino, "Owl : Congestion control with partially invisible networks via reinforcement learning," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2021, pp. 1–10.
[44] T. Gilad, N. Rozen-Schiff, P. B. Godfrey, C. Raiciu, and M. Schapira, "MPCC: Online learning multipath transport," in *Proc. 16th Int. Conf. Emerg. Netw. Exp. Technol. (CoNEXT)*, 2020, pp. 121–135.
[45] C. Liu, M. Xu, Y. Yang, and N. Geng, "DRL-OR: Deep reinforcement learning-based online routing for multi-type service requirements," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2021, pp. 1–10.
[46] Y. Guan, Y. Zhang, B. Wang, K. Bian, X. Xiong, and L. Song, "PERM: Neural adaptive video streaming with multi-path transmission," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Jul. 2020.
[47] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[48] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[49] *NS-3 Simulator*. Accessed: May 2022. [Online]. Available: https://www.nsnam.org/
[50] S. Ferlin, O. Alay, O. Mehani, and R. Boreli, "BLEST: Blocking estimation-based MPTCP scheduler for heterogeneous networks," in *Proc. IFIP Netw. Conf. (IFIP Networking) Workshops*, May 2016, pp. 431–439.

Jiangping Han (Member, IEEE) received the B.S. and Ph.D. degrees from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), China, in 2016 and 2021, respectively. From November 2019 to October 2021, she was a Visiting Scholar at the School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, AZ, USA. She is currently a Post-Doctoral Fellow and a Post-Doctoral Researcher with the School of Cyber Science and Technology, USTC. Her research interests include future internet architecture design and transmission optimization.

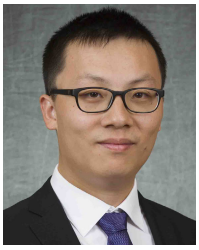
Kaiping Xue (Senior Member, IEEE) received the bachelor's degree from the Department of Information Security, University of Science and Technology of China (USTC), China, in 2003, and the Ph.D. degree from the Department of Electronic Engineering and Information Science (EEIS), USTC, in 2007. From May 2012 to May 2013, he was a Post-Doctoral Researcher with the Department of Electrical and Computer Engineering, University of Florida. He is currently a Professor with the School of Cyber Science and Technology, USTC. His research interests include next-generation internet architecture design, transmission optimization, and network security. He is a fellow of IET. He serves on the Editorial Board of several journals, including the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING (TDSC), the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS (TWC), and the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT (TNSM). He has also served as a (Lead) Guest Editor for many reputed journals/magazines, including IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS (JSAC), *IEEE Communications Magazine*, and IEEE NETWORK.

Jian Li (Member, IEEE) received the bachelor's degree from the Department of Electronics and Information Engineering, Anhui University, in 2015, and the Ph.D. degree from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), China, in 2020. From November 2019 to November 2020, he was a Visiting Scholar with the Department of Electronic and Computer Engineering, University of Florida. From December 2020 to December 2022, he was a Post-Doctoral Researcher with the School of Cyber Science and Technology, USTC, where he is currently an Associate Researcher. His research interests include wireless networks, next-generation internet, and quantum networks.

Rui Zhuang (Graduate Student Member, IEEE) received the B.S. degree from the Department of Information Engineering, Hefei University of Technology (HFUT), in July 2019. She is currently pursuing the Ph.D. degree with the School of Cyber Science and Technology, University of Science and Technology of China (USTC), China. Her research interests include future internet architecture design, transmission optimization, and data center networks.



Ruidong Li (Senior Member, IEEE) received the bachelor's degree in engineering from Zhejiang University, China, in 2001, and the Ph.D. degree in engineering from the University of Tsukuba in 2008. He is currently an Associate Professor with the College of Science and Engineering, Kanazawa University, Japan. Before joining Kanazawa University, he was a Senior Researcher with the Network System Research Institute, National Institute of Information and Communications Technology (NICT). His current research interests include future networks, big data networking, blockchain, information-centric networks, the Internet of Things, network security, wireless networks, and quantum internet. He is a member of IEICE, the Founder and the Chair of the IEEE SIG on big data intelligent networking and the IEEE SIG on intelligent internet edge, and the Secretary of the IEEE Internet Technical Committee. He also serves as the Chair for IWQoS 2021, MSN 2020, BRAINS 2020, ICC 2021 NMIC Symposium, ICCN 2019/2020, and NMIC 2019/2020. He organized the Special Issues for the *IEEE Communications Magazine*, the IEEE NETWORK, and the IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING (TNSE).



Ruozhou Yu (Senior Member, IEEE) received the Ph.D. degree in computer science from Arizona State University, Tempe, AZ, USA, in 2019. He is currently an Assistant Professor of computer science with North Carolina State University, Raleigh, NC, USA. His research interests include quantum networking, edge computing, network algorithms and optimization, distributed learning, and security and privacy. He received the U.S. NSF CAREER Award in 2021. He has served on the organizing committees for IEEE INFOCOM from 2022 to 2023 and IEEE IPCCC from 2020 to 2023, the TPC Track Chair for IEEE ICCCN 2023, and a TPC Member for IEEE INFOCOM from 2020 to 2023 and ACM Mobihoc in 2023.

Guoliang Xue (Fellow, IEEE) received the Ph.D. degree in computer science from the University of Minnesota in 1991. He is currently a Professor of computer science and engineering with Arizona State University, Tempe, AZ, USA. His research interests include QoS provisioning, machine learning, wireless networking, and the IoT. He received the IEEE Communications Society William R. Bennett Prize in 2019. He has served as the TPC Chair for IEEE INFOCOM'2010, IEEE Globecom'2020, and IEEE IWQoS'2021, and the General Chair for IEEE CNS'2014 and IEEE/ACM IWQoS'2019. He has served on the TPC of many conferences, including ACM CCS, ACM MOBIHOC, IEEE ICNP, and IEEE INFOCOM. He is the Steering Committee Chair of IEEE INFOCOM. He has served on the Editorial Board for IEEE/ACM TRANSACTIONS ON NETWORKING and an Area Editor for IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, overseeing 12 editors in the wireless networking area.

Qibin Sun (Fellow, IEEE) received the Ph.D. degree from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), in 1997. From 1996 to 2007, he was with the Institute for Infocomm Research, Singapore, where he was responsible for industrial and academic research projects in the area of media security and image and video analysis. He was also the Head of Delegates of Singapore in ISO/IEC SC29 WG1 (JPEG). He worked as a Research Scientist at Columbia University from 2000 to 2001. He is currently a Professor with the School of Cyber Science and Technology, USTC. He has published more than 120 papers in international journals and conferences. His research interests include multimedia security, network intelligence, and security.