Reliable Cybertwin-Driven Concurrent Multipath Transfer With Deep Reinforcement Learning

Chengxiao Yu, Wei Quan, *Member, IEEE*, Deyun Gao, *Senior Member, IEEE*, Yuming Zhang, Kang Liu, Wen Wu, *Member, IEEE*, Hongke Zhang, *Fellow, IEEE*, and Xuemin Shen, *Fellow, IEEE*

***Abstract*—It is well known that concurrent multipath transfer (CMT) can improve the transmission rate. However, due to multiple heterogeneous paths from users to the access network, a large number of out-of-order packets significantly degrade the overall transmission reliability. Cybertwin provides a potential solution to alleviate the packet out-of-order problem by accurately detecting and perceiving the path state. In this article, we investigate the data scheduling problem and propose a learning-based cybertwin-driven CMT algorithm to obtain the optimal data scheduling policy. In particular, we first formulate the data scheduling problem as an integer linear programming by taking the QoS metrics into account. To cope with the packet out-of-order problem in CMT, we propose a reliable cybertwin-CMT with deep reinforcement learning (CMT-DRL) algorithm to determine the data scheduling decisions. The proposed algorithm takes multipath throughput, end-to-end delay, and packet loss rate into account. Besides, CMT-DRL adopts an asynchronous learning framework to efficiently execute data collection, packet scheduling, and neural network training in sequence by decoupling model training and execution. We conduct extensive experiments in a P4-based programmable network platform. Experimental results indicate that the CMT-DRL outperforms the existing benchmarks in terms of the number of out-of-order packets, round-trip time, and throughput.**

***Index Terms*—Concurrent multipath transfer (CMT), cybertwin-driven, deep reinforcement learning (DRL).**

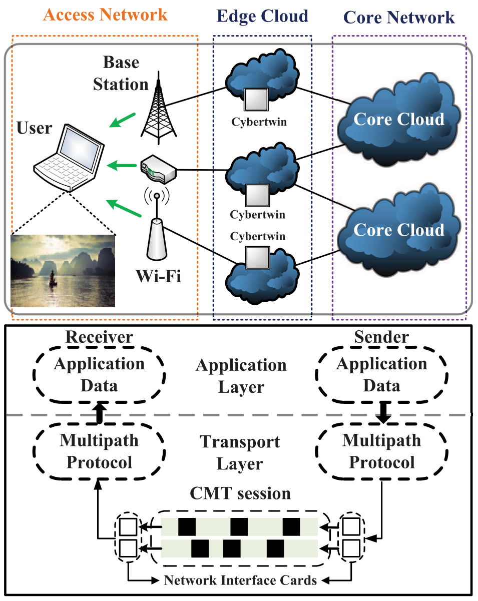


Fig. 1. Cybertwin-driven CMT architecture.

I. Introduction

With the emergence of bandwidth-hungry applications, more and more mobile terminals (i.e., tablet PCs and smartphones) have been equipped with multiple network interfaces, such as cellular networks and Wi-Fi [1]. Recently, cloud-computing-centric network architecture is provided to handle the big data explosion from Internet-of-Everything (IoE) devices. A cybertwin [2]-based next-generation network architecture (illustrated in Fig. 1) is proposed to perceive the network state in the future network. Besides, a novel network architecture, named SINET, is designed to remove the restrictions from the triple bindings completely in the conventional Internet, including resource/location binding, user/network binding, and control/data binding [3].

A large number of applications and services make full use of multiple paths to provide reliable transmission in the next-generation network [4], [5]. The wireless network environment needs to incorporate heterogeneous access networks to guarantee high-quality service provisioning [6], [7]. As a result, the multipath protocols have been proposed to mitigate throughput fluctuation to overcome high packet loss and link failure by load balance and resilience. The concurrent multipath transfer (CMT) [8] is considered as the most popular scheme to improve traffic transmission [9]. Such multipath transport protocols try to enhance the ability of path transfer by segmenting data traffic into separate paths. Multipath TCP (MPTCP) is a transport layer protocol, which uses multiple paths to transfer a single data stream [10], [11]. Additionally, it has

Manuscript received February 27, 2021; revised May 15, 2021; accepted July 14, 2021. Date of publication July 30, 2021; date of current version November 5, 2021. This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFE0206800; in part by the National Natural Science Foundation of Beijing, China, under Grant 4212010; and in part by the Fundamental Research Funds for the Central Universities under Grant 2019YJS021. *(Corresponding author: Wei Quan.)*

Chengxiao Yu, Wei Quan, Deyun Gao, Yuming Zhang, and Kang Liu are with the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China (e-mail: chengxiaoyu@bjtu.edu.cn; weiquan@bjtu.edu.cn; gaody@bjtu.edu.cn; 14111027@bjtu.edu.cn; 19111028@bjtu.edu.cn).

Wen Wu and Xuemin Shen are with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L3G1, Canada (e-mail: w77wu@uwaterloo.ca; sshen@uwaterloo.ca).

Hongke Zhang is with the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China, and also with the Research Center of Networks and Communications, Peng Cheng Laboratory, Shenzhen 518040, China (e-mail: hkzhang@bjtu.edu.cn).

Digital Object Identifier 10.1109/JIOT.2021.3101447

been confirmed that the stream control transmission protocol (SCTP) [12] is completely adapted to CMT with its multihoming property [13]. Therefore, CMT is a promising solution for multimedia applications against severe packet loss, high delay and low throughput network environments [14], [15].

However, the traditional CMT has suffered several issues caused by dynamic wireless network environments. We find that the complete gain from bandwidth aggregation of multiple paths is still far from an ideal situation [16], [17]. Since the different network characteristics of multiple paths, packets with a lower sequence number sent on a shorter path are more likely to reach the receiver. The receiver needs to retain lots of out-of-order packets, which cause path congestion in wireless networks [18], [19]. Therefore, classic heuristic data scheduling algorithms are throttled by a limited buffer space and tough packet out-of-order phenomenon. Many works focus on solving these problems to optimize data scheduling in multipath transfer networks [20].

Meanwhile, the high packet loss rate (PLR) is also an important concerning problem, which seriously decreases the end-to-end throughput in wireless networks. Traditional wireless technologies, such as high-speed rail communications, also suffer from high packet loss due to frequent handoff. Li *et al.* [21] observed that the PLR is higher than 20% during a period time of TCP stream when the train moves at a steady high speed. Zhang *et al.* [22] studied the Beijing–Tianjin intercity high-speed railway in which trains moving at a speed of about 300 km/h. They found that at scenarios with high speed, average throughput was much lower than that in the static scenario, and TCP throughput fluctuated greatly. Besides, if the sender transfers any lost packet, the sender will reduce the throughput and decrease channel utilization to achieve reliable data transfer, which dramatically decreases user experience [23], [24]. Therefore, it is imperative to solve the problem of high data volume and high-quality transmission in mobile scenarios. Fortunately, deep reinforcement learning (DRL) provides an opportunity to address this problem. Since DRL is an advantaged technique for data scheduling, we choose DRL as the tool for our system, which does not rely on accurate and mathematically solvable system models. Also, DRL can deal with a highly dynamic environment with time-varying user demands.

In this article, we investigate the data scheduling problem and propose a learning-based cybertwin-driven CMT algorithm to obtain the optimal data scheduling policy. First, we design a programming protocol-independent packet processors (P4)[1]-based CMT to transmit the data packets without deploying any protocol. Second, we formulate the data scheduling problem as an integer linear programming (ILP) by taking QoS characteristics into account. To cope with the packet out-of-order problem in CMT, a model-free DRL algorithm called CMT with deep reinforcement learning (CMT-DRL) is proposed to determine the data scheduling decisions. To speed up the training efficiency, we further propose an asynchronous training framework by decoupling the model training and execution. Specifically, the framework separates the whole training process into three phases: 1) data collection (i.e., the preparation phase); 2) packet scheduling (i.e., the online decision phase); and 3) neural network training (i.e., the offline training phase). After training the model using the real network data in an offline manner, the neural network parameters are updated to the online data scheduling policy in time. The proposed algorithm utilizes DRL's feature to learn to take the best action according to runtime states without relying on any predefined control policy.

[1]P4: https://p4.org/.

We implement the cybertwin-driven CMT-DRL algorithm through P4 language [25] based on the Linux system and compare its performance with the existing benchmarks. Extensive experimental results indicate that the CMT-DRL algorithm completely outperforms other data scheduling algorithms under different network conditions and scenarios. The main contributions of this article are summarized as follows.

1) We formulate the data scheduling problem as an ILP problem to acquire a comprehensive reward function by taking QoS metrics into account. We regard the process of packet transmission as a data scheduling problem.
2) We propose a reliable cybertwin-driven CMT-DRL algorithm to improve the data scheduling process by taking different network parameters of multiple paths into account. Besides, we design an asynchronous training framework, which decouples the model training and execution to efficiently carry out data collection, packet scheduling, and neural network training.
3) We implement the CMT-DRL algorithm using the P4 language in the Linux system. Experimental results demonstrate that the CMT-DRL algorithm outperforms the existing benchmarks in terms of the number of out-of-order packets, round-trip time (RTT), and throughput.

The remainder of this article is organized as follows. Section II presents the related work. Section III describes the problem state and formulation of cybertwin-driven CMT. The CMT-DRL algorithm, including programmable CMT mechanism and asynchronous learning framework is introduced in Section IV. Experimental results are presented in Section V to evaluate the advantage of CMT-DRL. Finally, we conclude this article in Section VI.

II. Related Work

A. Concurrent Multipath Transfer

Recently, many researchers pay much attention to the CMT algorithm to improve wireless network performance. Both MPTCP and SCTP have powerful multihoming functions and easily provide multipath transmission establishment to support CMT. Some research works have made contributions to CMT standardization in IETF in terms of SCTP. Meanwhile, compared to SCTP, MPTCP is expected to be backward compatible with conventional TCP and work with the existing network components such as middle boxes [26].

As previously mentioned, there are many works in the literature devoted to estimating link parameters to improve CMT solution performance. Wu *et al.* [27] investigated the key issue of limited channel resources and proposed a novel distortion-aware CMT scheme using mathematical formulation to support video streaming over heterogeneous wireless networks. Xu *et al.* [28] proposed a quality-aware adaptive concurrent multipath transmission (CMT-QA) scheme, which uses the estimated path quality as a benchmark for transmission scheduling. Another line of works focuses on optimizing CMT data scheduling. Dong *et al.* [29] proposed a new MPTCP scheduler, called LAMPS. The scheduler considers loss and delay when selecting subflows, and selects segments according to the state of the subflows. Besides, to reduce the path latency, Shi *et al.* [18] proposed and implemented a new scheduler, named STMS, which preallocated packets to send over the fast path for in-order packet arrival. The proposed algorithm can effectively alleviate the issue due to the host buffer size and in-network buffer size. In addition, Gao *et al.* [30] proposed a novel stochastic optimal scheduler for MPTCP that utilized the Lyapunov optimization technique in a software-defined wireless network. Nonetheless, the aforementioned solutions only consider partial network parameters to drive data scheduling, which is difficult to adapt to the various QoS requirements in highly dynamic network environments.

B. DRL in Networking

With the rapid development of DRL, it has been widely used in the field of network communications, such as the Internet of Vehicles (IoV) [31], [32], CMT data scheduling, and mobile-edge computing (MEC) [33]. Zhang *et al.* [34] studied the cooperative caching mechanism using a DRL approach to optimize the cooperation among edge servers. A network slicing-based architecture of NGWNs with artificial intelligence was proposed by Shen *et al.* [35] to address the network heterogeneity, dynamic environment, and diversified service requirements issues. Ning *et al.* [36] investigated the issues of edge computing and caching in IoV, which formulated a joint optimization problem to maximize MNO's profits DRL.

Besides, DRL is also used in CMT scenarios. For example, a RELES algorithm-based DRL was presented by Zhang *et al.* [37] for generating data scheduling policies to improve the network performance. The reward function considered some QoS metrics to optimize packet scheduling. A novel DRL-based algorithm called SmartCC was proposed by Li *et al.* [38] to address the multipath congestion control algorithm in heterogeneous networks by adopting a hierarchical tile coding algorithm.

Our work differs from the above works since we concentrate on improving data scheduling process using DRL as a tool by taking different network parameters into account. Besides, we realize the CMT mechanism using P4 language without any protocol, which executes comparison experiments.

III. Problem Statement and Formulation

A. Problem Statement

For the CMT data scheduling, especially in the heterogeneous networks, different data scheduling policies usually cause different performance variations. For example, some QoS metrics have extremely a huge difference in terms of multiple paths in the heterogeneous networks. When the network operator deploys the data scheduling policy according to the current network condition, it takes throughput, delay, and PLR from multiple paths into consideration. The data scheduling policies usually have conflict with these QoS metrics from multiple paths, which has a negative impact on the whole network performance. Network congestion will occur after several unsuitable data scheduling policies.

The CMT data scheduling policy takes packets from applications and determines which path to transmit each packet. We regard the process of packet transmission as a data scheduling problem. A time slot is defined as the duration of packets that are pushed into the multiple paths. At each time slot, a task selects the suitable path to transmit the data packets. This process continues until the data packets reach their destinations.

Considering a network topology with multiple paths in set $\mathcal{P} = \{P_1, P_2, P_3, \ldots, P_N\}$, where P_i represents the ith path of the network. Here, $\mathcal{N} = \{1, 2, \ldots, N\}$ denotes the number of multiple paths. We define some network characteristics of each path $\mathcal{P}_{\text{Tput}}$, $\mathcal{P}_{\text{Delay}}$, and $\mathcal{P}_{\text{PLR}}$ as the sets of throughput, delay, and PLR, respectively, with $\mathcal{P} = \mathcal{P}_{\text{Tput}} \cup \mathcal{P}_{\text{Delay}} \cup \mathcal{P}_{\text{PLR}}$. Therefore, the state of the multiple paths can be presented by a three tuple $\{\mathcal{P}_{\text{Tput}}, \mathcal{P}_{\text{Delay}}, \mathcal{P}_{\text{PLR}}\}$, where $\mathcal{P}_{\text{Tput}} = \{P_i^{\text{Tput}} | i \in \mathcal{N}\}$, $\mathcal{P}_{\text{Delay}} = \{P_i^{\text{Delay}} | i \in \mathcal{N}\}$, and $\mathcal{P}_{\text{PLR}} = \{P_i^{\text{PLR}} | i \in \mathcal{N}\}$.

B. Problem Formulation

Based on the above assumption, the data scheduling problem can be formulated as an ILP optimization problem with an objective of maximizing the long-term performance.

In addition, time is slotted into several periods in one time slot, denoted by set $\mathcal{T} = \{0, 1, 2, \ldots, T\}$, where T represents the duration of a data scheduling process. Therefore, at time τ, we use $P_i(\tau) = \{P_i^{\text{Tput}}(\tau), P_i^{\text{Delay}}(\tau), P_i^{\text{PLR}}(\tau)\}$ to describe the metrics of the ith path. For each path i, the number of data packets sending by system is denoted by $x_i(\tau)$, where $X(\tau) = \{x_i(\tau) | i \in \mathcal{N}, \tau \in \mathcal{T}\}$.

Let $d_i(\tau)$ denote the number of packets arriving at the ith path after sending from the source at time τ, where $D(\tau) = \{d_i(\tau) | i \in \mathcal{N}, \tau \in \mathcal{T}\}$. To estimate the performance of multiple paths, we use $d_i(\tau)/x_i(\tau)$ to denote the successful transmission rate. Then, we use BW_i to denote the bandwidth of the ith path. Therefore, the real throughput of the ith path can be presented by

$$P_i^{\text{Tput}}(\tau) = BW_i \frac{d_i(\tau)}{x_i(\tau)}. \tag{1}$$

Therefore, the total long-term average throughput of multiple paths can be presented by

$$P_{\text{Tput}} = \frac{1}{T} \sum_{\tau=0}^{T} \sum_{i=1}^{N} BW_i \frac{d_i(\tau)}{x_i(\tau)}. \tag{2}$$

In addition to the estimation of throughput, delay of each path is considered as an essential metric of the CMT path selection. The variation trend of delay lies on the RTT of each path. Therefore, the long-term average delay of multiple paths can be presented by

$$P_{\text{Delay}} = \frac{1}{T}\sum_{\tau=0}^{T}\sum_{i=1}^{N} P_i^{\text{Delay}}(\tau)x_i(\tau). \tag{3}$$

The last characteristic of multiple paths is PLR. Let $1 - d_i(\tau)/x_i(\tau)$ denote the PLR of the ith path at time τ, which is the ratio between the number of lost packets and the total number of sending packets. The long-term average PLR can be denoted by

$$P_{\text{PLR}} = \frac{1}{T}\sum_{\tau=0}^{T}\sum_{i=1}^{N}\left(1 - \frac{d_i(\tau)}{x_i(\tau)}\right). \tag{4}$$

The goals that we want to achieve should consider the following metrics.

1) *Throughput:* Throughput is the extremely important factor we should concern. The purpose of CMT data scheduling is to aggregate the multiple paths throughput and provide the maximal bandwidth to networks.
2) *Delay:* To address the issue of path diversity, it is necessary to reduce the delay of multiple paths to reduce the packet out-of-order rate and receiver buffer.
3) *PLR:* Low PLR can improve the performance of multiple paths.

Therefore, if there are N candidate paths in the CMT network, the objective function can be formulated by

$$R = aP_{\text{Tput}} - bP_{\text{Delay}} - cP_{\text{PLR}} \tag{5}$$

where a, b, and c are determined by the state of multiple paths based on its data scheduling policy and path diversity. Thus, the data scheduling problem with the objective of maximizing is formulated as follows:

$$\max \; R \tag{6}$$

$$\begin{aligned} \text{s.t.}\; & P_i^{\text{Delay}}(\tau) \le \bar{d} \\ & 1 - \frac{d_i(\tau)}{x_i(\tau)} \le \bar{p} \\ & 0 \le d_i(\tau) \le x_i(\tau) \\ & \forall i \in \mathcal{N} \quad \forall \tau \in \mathcal{T} \quad \forall d_i(\tau) \quad \forall x_i(\tau) \in \mathbb{N}^+ \end{aligned} \tag{7}$$

where $\bar{b}$ and $\bar{p}$ represent the threshold values of delay and PLR. $P_i^{\text{Delay}}(\tau) \le \bar{d}$ and $1 - ([d_i(\tau)]/[x_i(\tau)]) \le \bar{p}$ constrain that delay and PLR are not allowed to exceed the threshold values.

Although the optimal data scheduling policy can be provided by the above ILP optimization problem, it could waste a large amount of computing space and time to execute. Fortunately, DRL provides opportunities to solve the optimization problem, as a DRL agent can learn to make suitable decisions by interacting with a complicated environment and automatically adjust its parameters to achieve the optimal policy. In the following, the CMT-DRL algorithm is proposed to address the data scheduling problem, in which the multiple transmission system learns to adjust the data scheduling policy according to the states and actions of the network.

IV. CMT-DRL-Based Programmable Data Planes

In this section, we present a novel programmable CMT forwarding mechanism through the P4 language, which can flexibly configure the data scheduling parameters using the programmable switch feature to forward packets.

A. Programmable CMT Forwarding Mechanism

To solve the problem of poor scalability caused by OpenFlow, Bosshart *et al.* [25] proposed a P4 and the corresponding forwarding model. With the help of data plane programming capabilities brought by P4, administrators can not only implement existing network device functions such as routers and firewalls but also easily support new protocols.

The programmable data model defines several structural elements to achieve full-line protocol independent and programmable characteristics [39], including: 1) *header:* the header is used to implement a customized packet header and set the packet parsing rules by declaring an ordered list of the field names and lengths; 2) *P4 parse:* The parser implemented in the P4 language can flexibly parse the header fields of the data packet; and 3) *match–action table:* the match–action table is the basic unit for the data plane to perform forwarding logic. When the action is executed, the first instance in the analytical representation will be updated.

As mentioned before, the CMT can effectively improve the reliability of wireless networks by aggregating the multiple access paths. In the traditional SDN architecture, the data packets can be scheduled by the control plane with match and action forwarding processing paradigm. However, both SCTP and MPTCP, which support the CMT feature, need to install their protocol on the operating system. Especially, in terms of MPTCP, the researchers need to update and compile the Linux kernel to install MPTCP, which significantly increases the deployment time and decreases the system flexibility. Therefore, a new protocol-independent programmable technology P4 is adopted in this article to implement the CMT. We introduce the multipath forwarding paradigms with Version 1.2.0 of $P4_{16}$.

The programmable CMT forwarding mechanism is shown in Fig. 2. To better realize the multipath forwarding function, we divide the programmable CMT mechanism into three states: 1) parsing state; 2) match–action state; and 3) forwarding state. The detailed definitions of three states are as follows.

1) *Parsing State:* The programmable CMT requires packet header information to check the validity of the state. The function of the parser is to extract the header field from the packet and parse it according to the packet header parsing diagram. The parsing process will be compiled by the compiler into a data packet header parsing diagram and configured on the parser.
2) *Match-Action State:* The programmable CMT requires the command to forward different packets from the control plane. Match action units are represented by P4, which is called a table. Match actions may include data values that can be written in the control plane and read in the data plane. Action is the main component that allows the control plane to dynamically influence the behavior of the data plane.

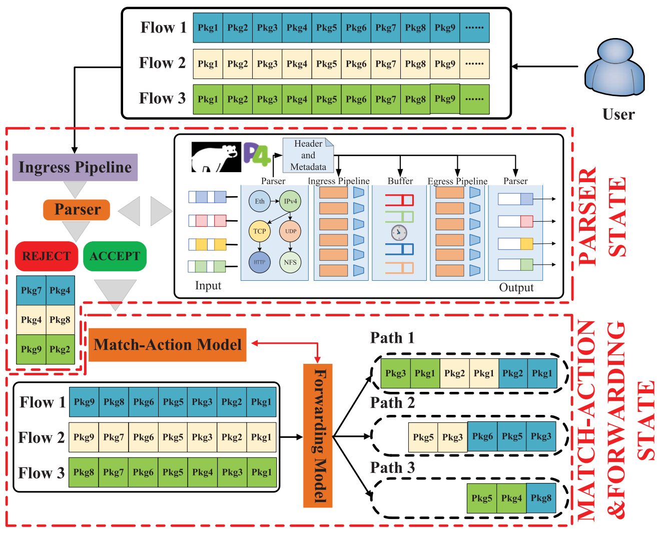


Fig. 2. Programmable CMT forwarding mechanism.

Algorithm 1 Programmable CMT Forwarding Mechanism

```
Input: $ingress\_port$: the next transfer port.
    $path\_threshold_i$: the number of transferred packets on the $i$-th path, $i \in [0, M]$
Output: $engress\_port$: the packets transfer ports.
1: Get the port from the current state,
   $temp\_port \leftarrow getport(current\_port)$;
2: Initialize $hash\_table \leftarrow \emptyset$, $temp\_count \leftarrow 0$;
3: if $packet\_state == valid$ then
4:     $hash\_table \leftarrow packet\_state$;
5:     while $i \leq M$ do
6:         if $temp\_count \leq path\_threshold_i$ then
7:             Update $temp\_count = temp\_count + 1$;
8:             Transfer the packet by $temp\_port$;
               $current\_port = temp\_port$;
9:             return $temp\_port$;
10:        else
11:            Update the $temp\_port = temp\_port + 1$;
               $i = i + 1$;
12:        end if
13:    end while
14: end if
```

3) *Forwarding State:* The programmable CMT requires per-flow forwarding state distributed by the flow table from control plane. The forwarding state determines the forwarding path according to the forwarding mechanism, which is determined by match-action state.

With these three states, the programmable CMT forwarding mechanism can utilize the programmable switch feature to forward the packets in the data plane. The traffic can be strictly managed by the programmable switch. Unless the priority traffic can be forwarded by rules, the first-in–first-out mechanism is adopted to transfer data packets. We now introduce the process of programmable CMT forwarding mechanism.

As shown in Algorithm 1, the programmable CMT forwarding mechanism can be described as follows.

1) First, considering different flows caused by different applications, if the packet is valid for extracting the header information, the tuple of the packet header information is converted to *hash_table* for different flows; otherwise, repeat the first step.
2) Second, we get *temp_port* from the programmable switch to establish the next transfer port. If *temp_count* is less than the current path threshold, the packets will be forwarded via the next port *temp_port*, and then we update *temp_count* to *temp_count* + 1.
3) Finally, if *temp_count* exceeds the current path threshold, the programmable CMT mechanism updates *temp_port* and forwards the packets into the next port until $path\_threshold \geq M$.

B. Deep Reinforcement Learning

RL is a type of machine learning algorithm. The problem of RL is the task of interaction between agent and environment, which determines action to maximize the expected reward in discrete decision steps. Therefore, there is a mapping between state and action, that is, a state can correspond to an action, or a probability set of taking different actions.

At each time slot, the agent receives state s_t in the current external interaction environment. According to policy $\pi(s)$, the agent chooses best action a_t in order to receive a reward r_t, where $a_t = \pi(s_t)$. To maximize the discounted cumulative reward $R_0 = \Sigma_{t=0}^{T}\gamma^t r(s_t, a_t)$, the agent selection policy $\pi(s)$ matches its state to an action or probability distribution, where $r(\cdot)$ is the reward function and the discount factor $\gamma \in [0, 1]$ determines the influence of following rewards compared with the current reward. If $\gamma = 0$, it considers maximizing the current reward. If γ approaches 1, the agent considers the long-term reward. The agent does not know what the optimal policy at the beginning. It often starts with a random policy to get a series of states, actions, and rewards. The RL model can be defined by a five-tuple $< S, \quad A, \quad R, \quad S', \quad T >$:

1) *S:* the possible state space. s_t is the specific state of S;
2) *A:* the possible action space. a_t is the specific state of A;
3) *R:* the reward space. r_t is the specific state of R;
4) *S′:* the possible next state space, s_{t+1} is the specific state of S';
5) *T:* the time slots of all states. t is the slot of one state to another.

However, it will take too many slots to receive the best policy, which is not suitable for large-scale networks. Recently, a breakthrough technique called deep learning [40] has been introduced. Therefore, a new technology for improving the shortcoming of RL called DRL is proposed. DRL makes use of deep neural networks (DNNs) to enhance the training ability, which can effectively accelerate the learning speed and the performance of an RL algorithm called deep Q networks (DQNs). A DQN takes a state-action pair (s_t, a_t) as input and outputs the corresponding $Q(s_t, a_t)$ value, which is the expected discounted cumulative reward

$$Q(s_t, a_t) = \mathbb{E}[R_t|s_t, a_t] \quad (8)$$

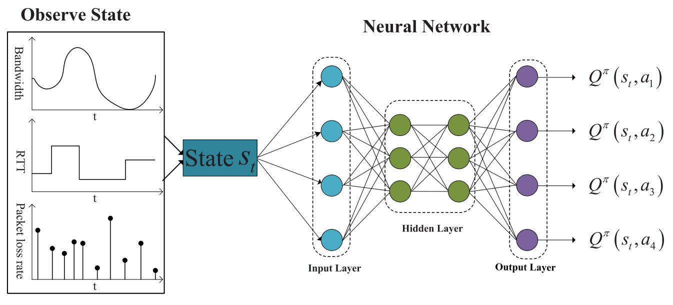


Fig. 3. DNN in the DQN.

where $R_t = \sum_{t=k}^{T} \gamma^k r(s_t, a_t)$. The action can be converted to a general policy:

$$\pi(s_t) = \arg\max Q(s_t, a_t). \tag{9}$$

Let t denote the tth time slot with the policy π. If the agent confirms that the optimal Q value maps to the best state, the policy will choose the action, which obtains the highest reward. The Q value obtained by the policy π can be represented by $Q^\pi(s_t, a_t)$. According to DQN, it is easy to get the Bellman equation to calculate $Q^*(s_t, a_t)$. Given action a_t in state s_t, the expected maximum Q-value can be expressed by

$$Q^*(s_t, a_t) = \mathbb{E}\left[r_t + \gamma \max_{a'} Q^\pi\left(s'_t, a'_t\right) | s_t, a_t\right]. \tag{10}$$

From (10), maximum $Q^*(s_t, a_t)$ and optimal data scheduling actions can be derived by the value and action iteration. The update process of $Q(s_t, a_t)$, namely, the Q-learning process can be expressed by

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \lambda\left[r_t + \gamma \max_{a'} Q^\pi\left(s'_t a'_t\right)\right]. \tag{11}$$

We utilize the feature of DNN to stand for the action-value function and address the problem of the large-scale state space. The relationship between state s_t and the value of each action a_t is shown in Fig. 3. The parameters in the DNN are denoted by θ. Then, the Q value function can be represented by

$$y_t = r(s_t, a_t) + \gamma Q\left(s_{t+1}, \pi(s_{t+1}) | \theta^Q\right) \tag{12}$$

where y_t is the target value of DQN. Meanwhile, DQN can be denoted by minimizing the loss of y_t

$$L\left(\theta^Q\right) = \mathbb{E}\left[\left(y_t - Q\left(s_t, a_t | \theta^Q\right)\right)^2\right]. \tag{13}$$

As mentioned in [41], it is known that the average reward decided by DRL is not stable because of the nonlinear function approximation. To address the problem, two mechanisms have been proposed.

1) *Experience Replay Mechanism:* Experience replay is mainly used to overcome the correlation and nonstationary distribution of training data. The DRL agent can generate and store many samples into the replay buffer for updating the DNN. The experience replay mechanism completely takes advantage of both new and old samples, then using a uniform random sampling method to extract transitions from the replay buffer for training neural networks.
2) *Target Network Mechanism:* The role of the target network is to disrupt correlation, which results in two networks with identical structures but different parameters in DRL. The independent target network approximates the main parameters of the network with a small amount of change each time. At the same time, the target network reduces the correlation between the current and the target Q values to a certain extent and improves the stability of the algorithm.

C. Reward Function

It is very important for us to utilize a single DRL agent (i.e., multipath data scheduler) to achieve the higher performance for all multiple paths. In order to realize the goal, a CMT-DRL architecture has been designed based on programmable data planes to maximize the overall utility of networks. We describe the state, action, and reward of CMT-DRL as follows.

Agent: An agent is an entity, which carries out the learning task in the system. In the CMT-DRL algorithm, the agent stands for the programmable scheduler, which generates data scheduling policies to separate the data into multiple paths in terms of different network environments.

State: A state of the system is the information of environment that can be observed by an agent. In CMT-DRL, we define the state based on the network parameters, i.e., delay, throughput, and PLR. Through network measurement tools, these parameters can be easily obtained. The state of a flow at a time slot is $s_t = (s_t^1, \ldots, s_t^i, \ldots, s_t^N)$, where s_t^i $(1 \le i \le N)$ is the observed state of the ith path; and N is the total number of available paths. Then, $s_t^i = (g_t^i, d_t^i, p_t^i)$ can be represented by the ith path performance, where:

1) g_t^i is the path throughput at time slot t;
2) d_t^i is the path average delay at time slot t;
3) p_t^i is the path PLR at time slot t.

Considering the impact on the end-to-end performance, we incorporate these key parameters into the state to improve training accuracy. Through extensive experiments, we discover that adding more unnecessarily parameters cannot improve network performance, which decreases the training efficiency.

Action: An action indicates how an agent responses on the observed state. In the CMT-DRL algorithm, the action is a packet distribution decision, which determines how to sperate the current packets into multiple paths. An action at the time slot is $a_t = (a_t^1, \ldots, a_t^i, \ldots, a_t^N)$, where $a_t^i(1 \le i \le N)$ represents that how many packets will be distributed into the ith path.

Reward: The reward is a scalar value. At each time slot, the environment returns a reward to the agent according to the action of the agent. The reward defines whether the behavior is good or bad in this situation, and the agent can adjust its policy according to the reward. At time slot t, according to state s_t, the agent takes an action a_t. After taking this action, the state will transit into s_{t+1}, and agent receives a reward. In this system, we aim to increase network performance and decrease the network delay and PLR of multiple transfer. Therefore, we take network metrics into consideration to maximize the performance. In time slot t, the state of agent is s_t, and action

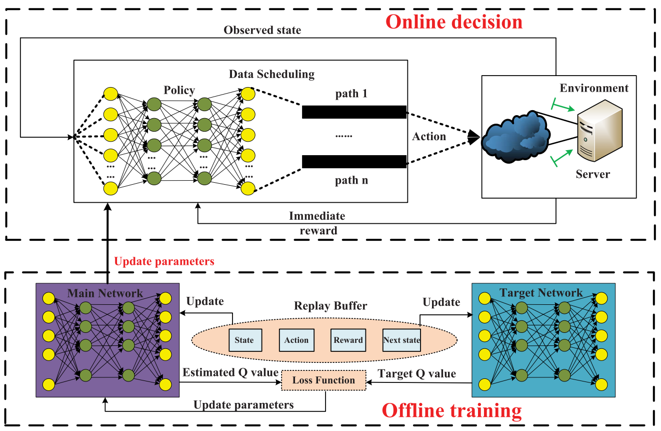


Fig. 4. Asynchronous training architecture.

a_t is took. The CMT-DRL algorithm uses the following reward to get better performance:

$$R(s_t, a_t) = aU_t^{\text{Tput}} - bU_t^{\text{Delay}} - cU_t^{\text{PLR}} \tag{14}$$

where $0 < a, b, c < 1$ are weights.

Here, $U_t^{\text{Tput}} = \sum_{i=0}^{N} g_t^i$ is the measured long-term total throughput of all TCP paths at one time slot. The value of the total throughput is a positive correlation with reward. $U_t^{\text{Delay}} = (1/N)\sum_{i=0}^{N} d_t^i$ and $U_t^{\text{PLR}} = (1/N)\sum_{i=0}^{N} p_t^i$ are the average delay and PLR of all TCP paths in one time slot, respectively, which is negative correlation to reward.

D. Asynchronous Learning Framework

In order to improve training efficiency, we propose an asynchronous training architecture based on data scheduling, which is shown in Fig. 4. Unlike other heuristic approaches that employ fixed scheduling algorithms, our algorithm tries to learn a data scheduling policy from network environment. The training process can be divided into three phases: 1) preparation phase; 2) online decision phase; and 3) offline training phase.

1) *Preparation Phase:* First, the agent does not know any policy because the policy table is empty. To enrich the replay buffer, the agent needs to collect data from the environment. In our training model, we first generate the random data scheduling policy to collect extensive experimental samples. Through these samples, the CMT-DRL will train the DNN parameters. After completing the experience collection, the offline training can generate the latest data scheduling policy.
2) *Offline Training Phase:* Because of the limited space and computing capability in the Linux kernel, an asynchronous training architecture called CMT-DRL is proposed, which separates the offline training from the whole training framework. In this phase, the agent uses the experience policy from the preparation phase to train the neural network parameters. The collected experience entries are five-tuple $(s_t, a_t, r_{t+1}, s_{t+1}, t)$ stored in replay buffer. The agent adopts these experience entries to update the neural network model.

Algorithm 2 CMT-DRL Algorithm

```
1:  / ** preparation phase ** /
2:  Initialize rule table $\mathbb{T} \leftarrow \emptyset$;
3:  Initialize replay buffer $\mathbb{B} \leftarrow \emptyset$;
4:  Apply the random data scheduling policy to generate an
    action $a_t$;
5:  Execute action $a_t$ and obverse $(s_t, a_t, r_{t+1}, s_{t+1}, t)$;
6:  Training Q-network weight $\theta^Q$ with $(s_t, a_t, r_{t+1}, s_{t+1}, t)$;
7:  Store transition sample $(s_t, a_t, r_{t+1}, s_{t+1}, t)$ into replay
    buffer $\mathbb{B}$;
8:  / ** offline training phase ** /
9:  Initialize target Q-network $Q'$ with weight $\theta^{Q'} \leftarrow \theta^Q$;
10: Initialize weight of with a random value $\tau \in (0, 1)$;
11: for $i = \{1, 2, \cdots, T\}$ do
12:     Sample some transitions randomly from $\mathbb{B}$;
13:     Set $y_t = r(s_t, a_t) + \gamma Q(s_{t+1}, \pi(s_{t+1})|\theta^Q)$;
14:     Update the parameter of neural network $\theta^Q$ by mini-
        mizing the loss:
        $L(\theta^Q) = \mathbb{E}[y_t - Q(s_t, a_t|\theta^Q)]$;
15:     Update the target network: $\theta^{Q'} \leftarrow \tau\theta^Q + (1 - \tau)\theta^{Q'}$;
16: end for
17: / ** online decision phase ** /
18: Initialize the policy network $\pi(s|\theta^\pi)$;
19: Synchronize policy $\theta^\pi$ to online neural network from
    offline training;
20: for $i = \{1, 2, \cdots, N\}$ do
21:     Receive initial observation state $s_t$;
22:     Match state $s_t$ with $a_t$ from $\mathbb{B}$ with $\epsilon$-greedy policy;
23:     Execute $a_t$ and obtain a transition $(s_t, a_t, r_{t+1}, s_{t+1}, t)$;
24:     Save the transition to $\mathbb{B}$;
25: end for
```

3) *Online Decision Phase:* Once the TCP connects, CMT-DRL may synchronize the rule table from experience entries to update the neural network model. During the offline training phase, the CMT-DRL observes the network state, and takes an action by looking up the policy table to obtain the data scheduling policy. In this phase, if the agent observe a state, which is not in the policy table, the agent may use Round-Robin (CMT-RR) algorithm instead of CMT-DRL. If the agent observes a state in the policy table, the agent will execute ϵ-greedy policy for exploration. With probability (1-ϵ), it takes the policy in the table, and with probability ϵ, it takes the random policy. The ϵ-greedy policy adopts here again due to the dynamic network environment.

Generally, we present an asynchronous CMT-DRL in Algorithm 2. Among all DRL methods, offline learning methods offer significant training efficiency compared with the online learning. Because of the frequently dynamic network environment, it is difficult for an RL model to response the network environment changes in time. Therefore, we train another neural network offline to detect changes under the same network conditions. Over time, we manually change the network conditions and use the converted samples as input to the neural network.

Our offline training phase is based on the DQN framework by extending the Q-learning action space. First, the algorithm randomly generates some transition samples to initialize all parameters θ^{π} of DNN. Then, we use target network y_t to improve the learning efficiency and stability. The parameter of neural network θ^{π} is updated by minimizing loss $L(\theta)$. Meanwhile, to remove the correlation among the subsequent training samples, the replay buffer offers the experience transition samples in the learning process. The learned experience transition samples $(s_t, a_t, r_{t+1}, s_{t+1}, t)$ at each step are stored in replay buffer $\mathbb{B}$. They are randomly selected to train the parameters of DQN with the ϵ-greedy policy for balancing exploration and exploitation.

In the online decision phase, the neural network will be synchronized by the online neural network from the offline training. During the execution phase, CMT-DRL observes the network status and receives the optimal policy for transmitting data packets through different paths by looking up the policy table. Because of the limited dimension of the rule table, the overhead of the table search time is negligible compared to the implementation time of multiple transmission strategies. Meanwhile, the collector receives the five-tuple from replay buffer $\mathbb{B}$ to update the policy network at the next stage.

V. Performance Evaluation

In this section, we evaluate the performance of CMT-DRL data scheduling algorithm on the programmable data planes. Particularly, we implement the CMT-DRL using the P4 language, which includes the client part and the server part.[2] We conduct extensive experiments to evaluate the multipath network performance in both homogeneous and heterogeneous scenarios. Three specified algorithms are also implemented to compare with CMT-DRL.

A. Experimental Setup

We analyze the CMT-DRL algorithm with some existing heuristic algorithms, including CMT-Random, CMT Round-Robin (CMT-RR), and CMT Weighted Round-Robin (CMT-WRR), which are some well-known data scheduling algorithms for CMT. We use the network simulator *Mininet* and the programmable switch. *Iperf3* is selected to evaluate the network performance. In our experiments, *Tcpdump* and *Wireshark* are used as the tools to capture and analyze all the data packets, respectively.

1) Network Topology: As shown in Fig. 5, we simulate a wireless network with two optional communication paths between the server and the client. Two terminals are used as a client and a server, respectively, which communicate with each other via two paths. Due to the decoupled training mode of CMT-DRL, there is no need to increase any device, like graphics processing unit (GPU), for the online decision making. Thus, the CMT-DRL can easily run and be trained on a regular laptop. The hardware parameter configuration of the experiment is presented in Table I. Under the current hardware configuration, we test the forwarding performance of the switch, whose maximum throughput is 1150 Mb/s.

[2]Codes for CMT-DRL are available via Github: https://github.com/ycx19930209/CMT-DRL.

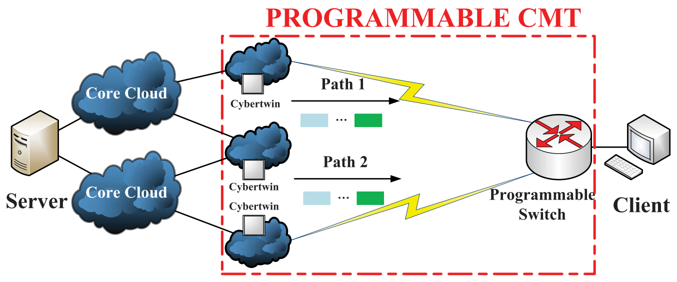


Fig. 5. Experimental topology.

TABLE I
HARDWARE CONFIGURATION

CPU	Intel Core i5-10210U CPU @1.60GHz 2.11GHz
RAM	16.0 GB
Storage	256 GB
Operating System	16.04.1-Ubuntu x86_64 GNU/Linux

In the preparation and offline training phases, we run CMT-DRL with various network parameter settings to learn the optimal data scheduling actions for multipath transfer. The following parameters settings are chosen through extensive experiments: reward function parameters: $a = 0.8$, $b = 0.1$, and $c = 0.1$; discount factor: $\theta = 0.99$; and $\epsilon = 0.01$ for the ϵ-greedy policy.

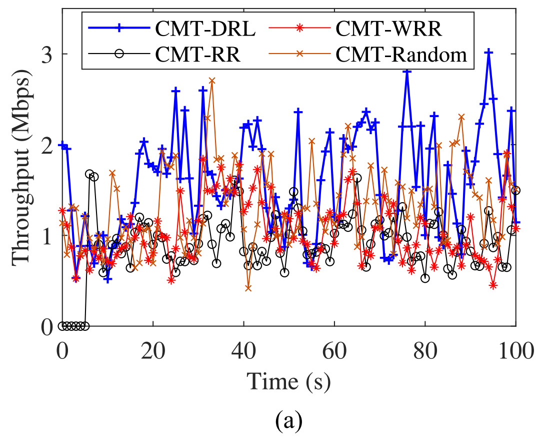
2) Comparison Algorithms: We select three heuristic algorithms as baselines: 1) CMT-Random; 2) CMT-RR; and 3) CMT-WRR. We create a python script to generate some data scheduling policies for the random algorithm. Meanwhile, we set the fixed data scheduling policy for the CMT-WRR algorithm according to the difference between two paths.

3) Performance Metrics: We choose the following metrics for CMT-DRL performance evaluation: 1) *application throughput:* the number of successfully transmitted bytes per unit time, which represents the network performance; 2) *application RTT:* the round trip time of a packet, which indicates the delay of the network; 3) *application PLR:* the PLR on one network, which stands for reliability of the network; and 4) *number of out-of-order packets:* the number of out-of-order packets reached in receivers, which declines network performance sharply.

B. Experimental Results

To comprehensively analyze the advantages of the CMT-DRL algorithm, both *homogeneous* and *heterogeneous* scenarios are considered in our evaluation. We evaluate the performance of CMT-DRL in two scenarios. TCP connection will be established between two programmable switches and be kept active during the test process. We use *Iperf3* to keep the network full of packets in the test network environment.

Homogeneous Scenario: In this test scenarios, we set the delay $d_1 = d_2 = 40$ ms, the bandwidth $b_1 = b_2 = 1$ Mb/s,



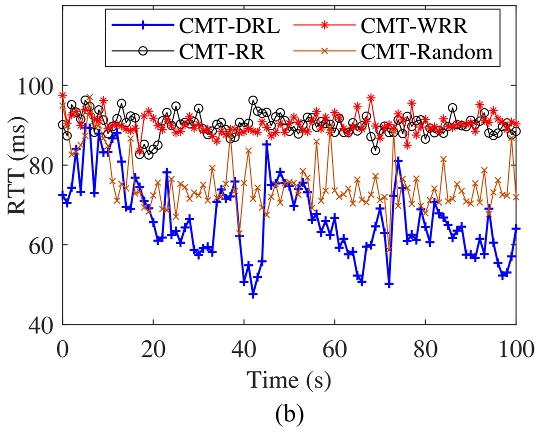
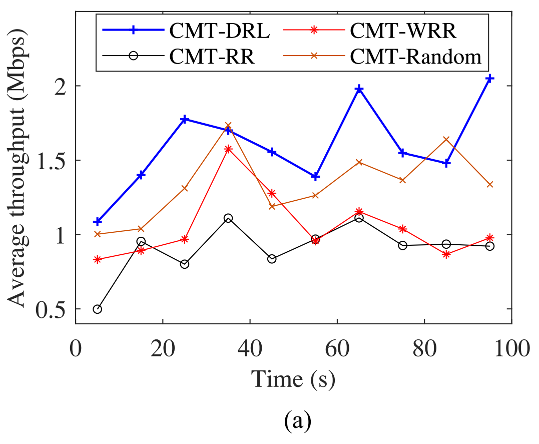


Fig. 6. Performance comparison in homogeneous scenario, in which $d_1 = d_2 = 40$ ms, $b_1 = b_2 = 1$ Mb/s, and $p_1 = p_2 = 2\%$. (a) Total throughput of multiple paths. (b) Total RTT of multiple paths.



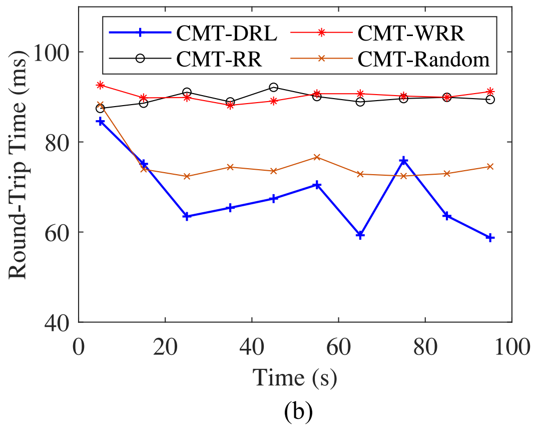


Fig. 7. Performance of average throughput and RTT in the homogeneous scenario. (a) Average throughput of multiple paths. (b) Average RTT of multiple paths.

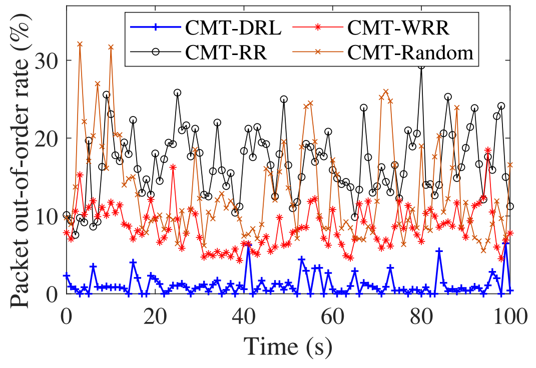


Fig. 8. Performance of packet out-of-order rate in the homogeneous scenario.

and the PLR $p_1 = p_2 = 2\%$ of two paths, respectively. Since both the bandwidths of the two paths are 1 Mb/s, the maximum bandwidth is 2 Mb/s if each path's bandwidth resources are fully utilized. We conduct extensive experiments and randomly select a sample data set, which lasts 100 s. The end-to-end performance of throughput and RTT for the mentioned four algorithms are shown in Fig. 6.

In Fig. 6(a) and (b), we present simulation results of throughput and RTT in this scenario. We can observe that the CMT-DRL algorithm outperforms other three algorithms in terms of throughput and RTT. As shown in Fig. 6(a), the throughput of CMT-DRL is much higher than other algorithms at most of time, and the average of its throughput fluctuates around 1.5 Mb/s, which can offer a stable network environment. Meanwhile, Fig. 6(b) illustrates the total RTT comparison of these algorithms. It is obvious that CMT-DRL algorithm achieves very good performance in terms of the end-to-end delay due to the deployment of the proposed asynchronous learning framework.

Besides, to compare the performance of various algorithms more intuitively, we make a statistic of the average throughput and RTT of each algorithm, which is illustrated in Fig. 7. As expected, both the average throughput and average RTT have the similar data characteristics and show the similar trend to that in Fig. 6. We observe that the average throughput of CMT-RR, CMT-WRR, and CMT-Random algorithms are only 0.93, 1.04, and 1.33 Mb/s, respectively, the average RTT of these three algorithms are about 90.22, 85.06, and 76.30 ms, respectively, which are much lower than ours.

On the other hand, we evaluate the packet out-of-order rate of all the four algorithms, as illustrated in Fig. 8. The simulation results show that the three benchmark algorithms' packet out-of-order rate is much higher than that of our proposed algorithm. Generally, compared with the CMT-Random, our algorithm improves the throughput by 18.8%, from 1.58 to 1.33 Mb/s; decreases the RTT by 8.9%, from 75.30 to 68.56 ms; and decreased the packet out-of-order rate from 13.09% to 1.11%. Therefore, our proposed algorithm can make

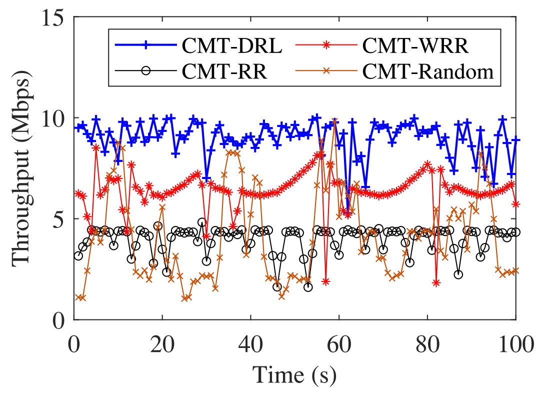


Fig. 9. Dynamic throughput of different algorithms in the heterogeneous scenario.

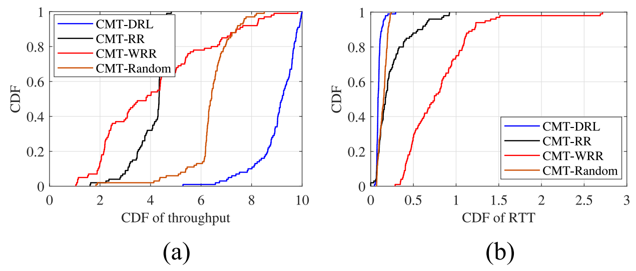


Fig. 10. CDF performance comparison of different algorithms in the heterogeneous scenario. (a) CDF throughput of multiple paths. (b) CDF RTT of multiple paths.

full use of bandwidth resource and dynamically allocate data packets to each path with an optimal scheduling policy.

Heterogeneous Scenario: In this scenario, we test the CMT-DRL algorithm in a heterogeneous environment. We set the two heterogeneous paths with different delay $d_1 = 50$ ms, $d_2 = 20$ ms, the bandwidth $b_1 = 2$ Mb/s and $b_2 = 8$ Mb/s, and the PLR $p_1 = 1\%$, $p_2 = 0.5\%$, respectively. The total throughput results of all algorithms are shown in Fig. 9. As observed, the CMT-RR algorithm has the lowest throughput whose average value is 3.99 Mb/s. CMT-DRL algorithm has a 9.01-Mb/s average throughput, which is the highest value among these algorithms. Meanwhile, the CMT-DRL algorithm has a smooth and steady tendency, whose values are between 7 and 10 Mb/s. Compared to the CMT-RR algorithm, CMT-DRL increases TCP throughput by almost two times.

We plot the cumulative distribution function (CDF) to compare the network performance of different algorithms in Fig. 10. Fig. 10(a) presents the performance of different algorithms on throughput and Fig. 10(b) presents on RTT. Similar to Fig. 9, the CMT-DRL algorithm outperforms the other three algorithms (i.e., CMT-RR, CMT-WRR and CMT-Random). Especially, compared to the CMT-RR algorithm, the CMT-DRL can further increase the performance in throughput in a heterogeneous scenario. It demonstrates that CMT-DRL can better adapt to different scenarios than the benchmarks by setting proper parameters in limited training episodes.

We also evaluate some other parameters via comparing the four algorithms. Fig. 11 depicts the experimental results, which

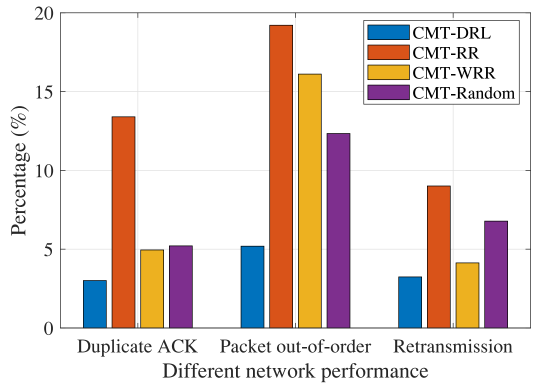


Fig. 11. Performance comparison of different algorithms in the heterogeneous scenario.

respectively, shows the duplicate ACK packet rate, the number of out-of-order packets rate and the timeout retransmission rate during the testing period. It is worth mentioning that CMT-DRL still performs better than the other three heuristic algorithms in a heterogeneous scenario. It is also observed that the CMT-RR algorithm has the highest throughput fluctuation due to the opposition of the algorithm itself in the heterogeneous scenario.

VI. Conclusion

In this article, we have investigated the data scheduling problem in CMT and proposed a reliable cybertwin-driven CMT with the DRL algorithm to flexibly determine the data scheduling policy. The proposed CMT-DRL algorithm can adjust the data scheduling policy according to cybertwin-based multipath behavior awareness. Besides, an asynchronous training architecture has been adopted to efficiently execute data collection, packet scheduling, and neural network training. We have implemented the proposed CMT-DRL using P4 paradigm and tested its performance compared to benchmark algorithms. Experimental results have demonstrated the advantages of the proposed algorithm.

In future work, we will further investigate the impact of congestion window size on data scheduling process in CMT. Moreover, we will take the congestion window size and the aforementioned three metrics together into account to improve the accuracy of data scheduling policies.

Acknowledgment

The authors would like to thank the reviewers for their valuable comments.

References

[1] N. Cheng *et al.*, "Space/aerial-assisted computing offloading for IoT applications: A learning-based approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1117–1129, May 2019.
[2] Q. Yu, J. Ren, Y. Fu, Y. Li, and W. Zhang, "Cybertwin: An origin of next generation network architecture," *IEEE Wireless Commun.*, vol. 26, no. 6, pp. 111–117, Dec. 2019.
[3] H. Zhang, W. Quan, H.-C. Chao, and C. Qiao, "Smart identifier network: A collaborative architecture for the future Internet," *IEEE Netw.*, vol. 30, no. 3, pp. 46–51, May/Jun. 2016.

[4] Y. Chen, Y. Zhang, Y. Wu, L. Qi, X. Chen, and X. Shen, "Joint task scheduling and energy management for heterogeneous mobile edge computing with hybrid energy supply," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8419–8429, Sep. 2020.

[5] S. Yao, Z. Li, J. Guan, and Y. Liu, "Stochastic cost minimization mechanism based on identifier network for IoT security," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 3923–3934, May 2020.

[6] H. Zhou, N. Cheng, J. Wang, J. Chen, Q. Yu, and X. Shen, "Toward dynamic link utilization for efficient vehicular edge content distribution," *IEEE Trans. Veh. Technol.*, vol. 68, no. 9, pp. 8301–8313, Sep. 2019.

[7] C. Chen, R. A. Berry, M. L. Honig, and V. G. Subramanian, "Pricing, bandwidth allocation, and service competition in heterogeneous wireless networks," *IEEE/ACM Trans. Netw.*, vol. 28, no. 5, pp. 2299–2308, Oct. 2020.

[8] P. Dong, B. Song, H. Zhang, and X. Du, "Improving onboard Internet services for high-speed vehicles by multipath transmission in heterogeneous wireless networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 12, pp. 9493–9507, Dec. 2016.

[9] I. A. Halepoto, N. H. Phulpoto, F. A. Jokhio, and A. R. Khatri, "Evaluation of multipath transmission using the stream control transmission protocol," *Int. J. Comput. Netw. Technol.*, vol. 5, no. 3, pp. 131–136, 2017.

[10] C. Yu, W. Quan, N. Cheng, S. Chen, and H. Zhang, "Coupled or uncoupled? Multi-path TCP congestion control for high-speed railway networks," in *Proc. IEEE/CIC ICCC*, 2019, pp. 612–617.

[11] C. Paasch, S. Ferlin, O. Alay, and O. Bonaventure, "Experimental evaluation of multipath TCP schedulers," in *Proc. ACM SIGCOMM Workshop Capacity Sharing Workshop*, 2014, pp. 27–32.

[12] T. D. Wallace and A. Shami, "Concurrent multipath transfer using SCTP: Modelling and congestion window management," *IEEE Trans. Mobile Comput.*, vol. 13, no. 11, pp. 2510–2523, Nov. 2014.

[13] X. Wang, M. Jia, Q. Guo, I. W.-H. Ho, and J. Wu, "Joint power, original bandwidth, and detected hole bandwidth allocation for multi-homing heterogeneous networks based on cognitive radio," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2777–2790, Mar. 2019.

[14] K. Xue *et al.*, "DPSAF: Forward prediction based dynamic packet scheduling and adjusting with feedback for multipath TCP in lossy heterogeneous networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 2, pp. 1521–1534, Feb. 2018.

[15] J. Wu, C. Yuen, B. Cheng, M. Wang, and J. Chen, "Energy-minimized multipath video transport to mobile devices in heterogeneous wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1160–1178, May 2016.

[16] M. Polese, F. Chiariotti, E. Bonetto, F. Rigotto, A. Zanella, and M. Zorzi, "A survey on recent advances in transport layer protocols," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3584–3608, 4th Quart., 2019.

[17] R. Sun *et al.*, "QoE-driven transmission-aware cache placement and cooperative beamforming design in Cloud-RANs," *IEEE Trans. Veh. Technol.*, vol. 69, no. 1, pp. 636–650, Jan. 2020.

[18] H. Shi *et al.*, "STMS: Improving MPTCP throughput under heterogeneous networks," in *Proc. USENIX ATC*, pp. 719–730.

[19] R. Sun *et al.*, "Delay-oriented caching strategies in D2D mobile networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 8529–8541, Aug. 2020.

[20] J. Wang, J. Liao, and T. Li, "OSIA: Out-of-order scheduling for in-order arriving in concurrent multi-path transfer," *J. Netw. Comput. Appl.*, vol. 35, no. 2, pp. 633–643, Mar. 2012.

[21] L. Li *et al.*, "A measurement study on multi-path TCP with multiple cellular carriers on high speed rails," in *Proc. ACM SIGCOMM*, 2018, pp. 161–175.

[22] Y. Zhang, P. Dong, S. Yu, H. Luo, T. Zheng, and H. Zhang, "An adaptive multipath algorithm to overcome the unpredictability of heterogeneous wireless networks for high-speed railway," *IEEE Trans. Veh. Technol.*, vol. 67, no. 12, pp. 11332–11344, Dec. 2018.

[23] H. Zhang, W. Quan, J. Song, Z. Jiang, and S. Yu, "Link state prediction-based reliable transmission for high-speed railway networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 12, pp. 9617–9629, Dec. 2016.

[24] C. Yu, W. Quan, S. Yu, and H. Zhang, "On the two time scale characteristics of wireless high speed railway networks," in *Proc. IEEE ICC*, 2017, pp. 1–6.

[25] P. Bosshart *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, 2014.

[26] C. Raiciu *et al.*, "How hard can it be? Designing and implementing a deployable multipath TCP," in *Proc. USENIX NSDI*, 2012, pp. 399–412.

[27] J. Wu, B. Cheng, C. Yuen, Y. Shang, and J. Chen, "Distortion-aware concurrent multipath transfer for mobile video streaming in heterogeneous wireless networks," *IEEE Trans. Mobile Comput.*, vol. 14, no. 4, pp. 688–701, Apr. 2015.

[28] C. Xu, T. Liu, J. Guan, H. Zhang, and G.-M. Muntean, "CMT-QA: Quality-aware adaptive concurrent multipath data transfer in heterogeneous wireless networks," *IEEE Trans. Mobile Comput.*, vol. 12, no. 11, pp. 2193–2205, Nov. 2013.

[29] E. Dong, M. Xu, X. Fu, and Y. Cao, "LAMPS: A loss aware scheduler for multipath TCP over highly lossy networks," in *Proc. IEEE 42nd Conf. LCN*, 2017, pp. 1–9.

[30] K. Gao, C. Xu, J. Qin, L. Zhong, and G.-M. Muntean, "A stochastic optimal scheduler for multipath TCP in software defined wireless network," in *Proc. IEEE ICC*, 2019, pp. 1–6.

[31] T. Fu, C. Wang, and N. Cheng, "Deep-learning-based joint optimization of renewable energy storage and routing in vehicular energy network," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6229–6241, Jul. 2020.

[32] W. Xu, S. Guo, S. Ma, H. Zhou, M. Wu, and W. Zhuang, "Augmenting drive-thru Internet via reinforcement learning-based rate adaptation," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3114–3123, Apr. 2020.

[33] J. Du, F. R. Yu, G. Lu, J. Wang, J. Jiang, and X. Chu, "MEC-assisted immersive VR video streaming over terahertz wireless networks: A deep reinforcement learning approach," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9517–9529, Oct. 2020.

[34] Y. Zhang *et al.*, "Cooperative edge caching: A multi-agent deep learning based approach," *IEEE Access*, vol. 8, pp. 133212–133224, 2020.

[35] X. Shen *et al.*, "AI-assisted network-slicing based next-generation wireless networks," *IEEE Open J. Veh. Technol.*, vol. 1, pp. 45–66, Feb. 2020.

[36] Z. Ning *et al.*, "Joint computing and caching in 5G-envisioned Internet of Vehicles: A deep reinforcement learning-based traffic control system," *IEEE Trans. Intell. Transp. Syst.*, early access, Feb. 5, 2020, doi: 10.1109/TITS.2020.2970276.

[37] 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 INFOCOM*, 2019, pp. 1648–1656.

[38] 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.

[39] *P4_16 Language Specification, Version 1.2.0*. Accessed: Oct. 23, 2019. [Online]. Available: https://p4.org/p4-spec/docs/P4-16-v1.2.0.html

[40] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.

[41] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

Chengxiao Yu was born in Dalian, Liaoning, China, in February 1993. He received the B.E. degree in communication and information systems from Beijing Jiaotong University, Beijing, China, in 2015, where he is currently pursuing the Ph.D. degree with the School of Electronic and Information Engineering, National Engineering Lab for Next Generation Internet Technologies.

His research interests include concurrent multipath transfer, machine learning, high-speed railway networks, and programmable data planes.

Wei Quan (Member, IEEE) received the Ph.D. degree in communication and information system from Beijing University of Posts and Telecommunications, Beijing, China, in 2014.

He is currently an Associate Professor with the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing. He has published more than 20 papers in prestigious international journals and conferences, including *IEEE Communications Magazine*, IEEE WIRELESS COMMUNICATIONS, IEEE NETWORK, the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE COMMUNICATIONS LETTERS, IFIP Networking, IEEE ICC, and IEEE GLOBECOM. His research interests include key technologies for network analytics, future Internet, 5G networks, and vehicular networks.

Dr. Quan is a TPC Member of IEEE ICC in 2017 and 2018, IEEE INFOCOM (NewIP Workshop) in 2020, ACM MOBIMEDIA in 2015, 2016, and 2017, and IEEE CCIS in 2015 and 2016. He serves as an Associate Editor for the *Journal of Internet Technology*, *Peer-to-Peer Networking and Applications*, and IEEE ACCESS, and as a technical reviewer for many important international journals. He is also a member of ACM and a Senior Member of the Chinese Association of Artificial Intelligence.

Deyun Gao (Senior Member, IEEE) received the B.E. and M.E. degrees in electrical engineering and the Ph.D. degree in computer science from Tianjin University, Tianjin, China, in 1994, 1999, and 2002, respectively.

He spent one year as a Research Associate with the Department of Electrical and Electronic Engineering, Hong Kong University of Science and Technology, Hong Kong. He then spent three years as a Research Fellow with the School of Computer Engineering, Nanyang Technological University, Singapore. In 2007, he joined the faculty of Beijing Jiaotong University, Beijing, China, as an Associate Professor with the School of Electronics and Information Engineering and was promoted to a Full Professor, in 2012. In 2014, he was a Visiting Scholar with the University of California at Berkeley, Berkeley, CA, USA. His research interests include the Internet of Things, vehicular networks, and the next-generation Internet.

Yuming Zhang received the B.S. degree in communication and information systems from Beijing Jiaotong University, Beijing, China, in 2014, where he is currently pursuing the Ph.D. degree in telecommunications and information system with the National Engineering Laboratory for Next Generation Internet Interconnection Devices.

His current research interests include software-defined networking, network function virtualization, mobile-edge computing, queue theory, and machine learning.

Kang Liu received the B.E. degree from Henan University of Science and Technology, Henan, China, in 2016. He is currently pursuing the Ph.D. degree with the School of Electronic and Information Engineering, National Engineering Lab for Next Generation Internet Technologies, Beijing Jiaotong University, Beijing, China.

His research interests include concurrent multipath transfer, machine learning, high-speed railway networks, and programmable data plane, wireless communication theory and technology, and wireless mesh networks.

Wen Wu (Member, IEEE) received the B.E. degree in information engineering from the South China University of Technology, Guangzhou, China, in 2012, the M.E. degree in electrical engineering from the University of Science and Technology of China, Hefei, China, in 2015, and the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2019.

Since 2019, he has been working as a Postdoctoral Fellow with the Department of Electrical and Computer Engineering, University of Waterloo. His research interest includes millimeter-wave networks and artificial intelligence-empowered wireless networks.

Hongke Zhang (Fellow, IEEE) received the M.S. and Ph.D. degrees in electrical and communication systems from the University of Electronic Science and Technology of China, Chengdu, China, in 1988 and 1992, respectively.

From 1992 to 1994, he was a Postdoctoral Fellow with Beijing Jiaotong University, Beijing, China, where he is currently a Professor with the School of Electronic and Information Engineering and the Director of a National Engineering Lab on Next Generation Internet Technologies. He is also with the Research Center of Networks and Communications, Peng Cheng Laboratory, Shenzhen, China. He has authored more than ten books and the holder of more than 70 patents. His research has resulted in many papers, books, patents, systems, and equipment in the areas of communications and computer networks.

Prof. Zhang is the Chief Scientist of a National Basic Research Program of China (973 Program) and has also served on the editorial board of several international journals.

Xuemin (Sherman) Shen (Fellow, IEEE) received the Ph.D. degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990.

He is currently a University Professor with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. His research focuses on network resource management, wireless network security, Internet of Things, 5G and beyond, and vehicular *ad hoc* and sensor networks.

Dr. Shen received the R.A. Fessenden Award in 2019 from IEEE, Canada, the Award of Merit from the Federation of Chinese Canadian Professionals (Ontario) in 2019, the James Evans Avant Garde Award in 2018 from the IEEE Vehicular Technology Society, the Joseph LoCicero Award in 2015 and the Education Award in 2017 from the IEEE Communications Society, and the Technical Recognition Award from Wireless Communications Technical Committee in 2019, and the AHSN Technical Committee in 2013. He has also received the Excellent Graduate Supervision Award in 2006 from the University of Waterloo and the Premiers Research Excellence Award in 2003 from the Province of Ontario. He served as the Technical Program Committee Chair/Co-Chair for IEEE Globecom'16 and IEEE Infocom'14, and the Chair for the IEEE Communications Society Technical Committee on Wireless Communications. He was the elected IEEE Communications Society Vice President for Technical and Educational Activities, the Vice President for Publications, the Member-at-Large on the Board of Governors, the Chair of the Distinguished Lecturer Selection Committee, and the Member of IEEE ComSoc Fellow Selection Committee. He was the Editor-in-Chief of the IEEE INTERNET OF THINGS JOURNAL, IEEE NETWORK, and *IET Communications*. He is a Registered Professional Engineer of Ontario, Canada, a Fellow of Engineering Institute of Canada, Canadian Academy of Engineering, and Royal Society of Canada, a Foreign Member of Chinese Academy of Engineering, and a Distinguished Lecturer of the IEEE Vehicular Technology Society and Communications Society.