Achieving Flexible and Lightweight Multipath Congestion Control Through Online Learning

Rui Zhuang, *Graduate Student Member, IEEE*, Jiangping Han, Kaiping Xue, *Senior Member, IEEE*, Jian Li, *Member, IEEE*, David S. L. Wei, *Life Senior Member, IEEE*, Ruidong Li, *Senior Member, IEEE*, Qibin Sun, *Fellow, IEEE*, and Jun Lu

***Abstract*—The upgrade of network devices to be equipped with multiple network interfaces makes it possible to improve network throughput performance through multipath transmission protocols, especially multipath TCP (MPTCP). However, so far the mostly used MPTCP protocols have a common limitation, namely the rigid and conservative method. They have been designed with little consideration of the fact that real networks are dynamic and the network status changes frequently, thus leading to the poor performance of current MPTCP in many realistic scenarios. In this paper, we propose a lightweight multipath congestion control algorithm based on online learning, named MP-OL. MP-OL models congestion control as a multi-armed bandit problem, and adjusts the sending rate of each subflow flexibly and adaptively through online learning. Therefore, MP-OL possesses the capability of suiting various network scenarios, and can achieve fairness and high performance in dynamic network environment. It can also flexibly switch between online learning and traditional method, which reduces the computational complexity while ensuring the learning efficiency, thus making MP-OL easy to deploy and use. As the experimental results demonstrated, compared with the leading MPTCP variants, MP-OL achieves significant improvements in fairness and link utilization, and shows better resilience to non-congestion loss and better adaptability to unstable network conditions. In real networks, MP-OL also obtains better throughput performance.**

***Index Terms*—Congestion control, multipath transmission, multipath TCP, online learning.**

I. Introduction

NOWADAYS most network devices are equipped with multiple network interfaces, which makes multipath capability an efficient way to improve the end-to-end transmission performance [1]. For example, most mobile devices, such as smartphones, are equipped with multi-radio interfaces (e.g., LTE and Wi-Fi), resulting in more and more multi-homed hosts accessing the Internet. Besides, data center networks are also motivated to adopt multipath transmission to make full use of the abundant network resources from the server domain. As an extension to TCP and a representative multipath transmission proposal, Multipath TCP (MPTCP) [2] has gained intense attention from both academia and industry because of its attractive characteristics.

Congestion control is an important and challenging issue in the design and implementation of MPTCP. In practice, the change of network condition is usually unpredictable, which causes the existing multi-path congestion control algorithms to face the challenges in heterogeneous and dynamic network environments. This is because the dynamic and uncertainty of a network make it hard for the control policies to find the relationship between its input and output. However, some leading MPTCP variants (e.g., Lia [3] and Olia [4], [5]) are ill-prepared for dealing with such a complex relationship as they adopt conservative predefined rules. Therefore, to deal with the lower-than-expected performance of current MPTCP and meet the increasing demands of applications, recent studies begin to use an efficient method, i.e., machine learning (ML), to learn such complex behaviors, so as to realize flexible high-performance multipath congestion control [6], [7], [8], [9].

The application of machine learning enables impressive accuracy improvements across many network control tasks (e.g., [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16]), and has attained outstanding achievements in the research of improving network performance. However, though there is a variety of great achievements, some problems that cannot be ignored are becoming prominent. We summarize these problems into two aspects: cost of research and development (R&D) and practicality.

First, the accuracy of most ML models depends on the consumption of computational resources. ML tasks belong to compute-intensive applications, whose main energy consumption is embodied in data analysis and computation. In the process of ML, it is necessary to continuously carry out real-time or offline classification and data analysis, which undoubtedly increases the complexity of the algorithm. As a result, substantial computational resources are needed to obtain an accurate and advanced model, which will consume considerable energy and generate huge financial and environmental costs.

Manuscript received 6 April 2022; revised 17 August 2022; accepted 5 September 2022. Date of publication 22 September 2022; date of current version 7 March 2023. This work is supported in part by the National Natural Science Foundation of China under Grant No. 61972371, and Youth Innovation Promotion Association of the Chinese Academy of Sciences (CAS) under Grant No. Y202093. The associate editor coordinating the review of this article and approving it for publication was L. Cui. *(Corresponding authors: Jiangping Han; Kaiping Xue.)*

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

David S. L. Wei is with the Department of Computer and Information Science, Fordham University, Bronx, NY 10458 USA.

Ruidong Li is with the College of Science and Engineering, Kanazawa University, Kanazawa 9201192, Japan.

Jun Lu is with the School of Cyber Science and Technology and the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei 230027, Anhui, China.

Digital Object Identifier 10.1109/TNSM.2022.3208747

Second, the schemes based on ML are difficult to be put into practical use. The main reason is two-fold: the disappointing performance of the model and the numerous difficulties in deployment. Specifically, the schemes that employ offline learning [6], [7], [8], [13], [14], [15], [16] usually assume that the behavior of the network has converged, and they require a pre-trained model before being practically used. Unfortunately, such assumption and method are flawed when applied to actual congestion control. First, training data is very important to the accomplishment of a successful ML model, while the reality is that it is difficult for most researchers to obtain sufficient and reliable data for training. Second, when the real network condition deviates from the assumption, the effectiveness of the model may have a serious dilution. Therefore, to ensure the reliability of models, ML-based schemes need to be constantly updated and maintained even after actual deployment, which increases users' burden and puts higher demands on users' equipment.

The above issues are built-in problems of off-line learning, which means the ML-based schemes that employ off-line learning could all suffer from these issues. And in the short term, such issues cannot be solved by some remedial means within the frame of the systems. On the contrary, the schemes that are designed completely based on online learning have some unique merits, which make them ideal for the Internet field. Online learning has significant advantages over offline learning. First, online learning can generate samples, train samples and give feedback in real-time, so it solves the problem of unreliable training data. And its training samples and results will also change with the changes of network environment, which makes them perfect for Internet applications with high requirements in real-time processing. Second, online learning is good at accurately estimating the environment [17]. Thus, compared with traditional algorithms, algorithms combined with online learning can support the decision-making under various kinds of network scenarios more effectively. Moreover, online learning allows flexibility and variety in the use of learning algorithms, which makes it possible to realize practical, flexible and lightweight intelligent congestion control algrithms.

To support the decision-making under dynamic and uncertain network environment, we design MP-OL, an online learning-based multipath congestion control algorithm. We also consider the deployability and feasibility of ML-based MPTCP congestion control algorithm. When an algorithm deploys machine learning, its consumption of computing resources and complexity may increase significantly [18], and multipath schemes will further multiply the consumption. For the sender of small mobile devices (e.g., smartphone), the high computational complexity and the parallel usage of multiple interfaces via MPTCP can rapidly consume the energy of mobile devices, thus affecting the experience of mobile users [19]. For the sender of mainframe servers, in the case of huge concurrency and high terminal load, the extra computing resources consumed by learning-based multipath scheme may slow down the system performance.

Therefore, for the long-term development, MP-OL is designed to be lightweight and low-complexity. We insert a switch module which combines high-performance online learning method with low-complexity traditional method to reduce the complexity of the algorithm. MP-OL only carries out online learning with low computational complexity at the right time for performance gains, rather than in the whole process of congestion control. To find the right opportunity for carrying out online learning, a method to identify the change of network state is also included.

The main contributions of this paper are as follows:

- We propose a phased online multipath congestion control algorithm, named MP-OL. To provide high learning efficiency as well as low complexity, MP-OL selects different rules for congestion control according to current network state, it can adapt to the dynamic state variety of the network and achieve high link utilization, thereby effectively improving the performance of MPTCP.
- We establish a model of multipath congestion control from an online learning perspective. This model is designed based on multi-armed bandit (MAB), and utilizes a utility function to quantify the benefit of each arm. It can make a quick and accurate understanding of the changing environment and converge fast to the optimal sending rate through consecutive exploration and exploitation.
- We implement MP-OL in the Linux kernel, and conduct extensive experiments in both of our laboratory platform and real network to evaluate its performance. Experimental results show that MP-OL achieves significant improvements in fairness and link utilization. Moreover, through testing in a variety of experimental environments, we prove that MP-OL has good generalization capability and can maintain satisfactory performance under unstable network conditions.

The rest of this paper is organized as follows: Some related works are introduced in Section II. In Section III, the system model is first presented, and the way to achieve valid congestion control from an online learning perspective is analyzed. After that, our multipath congestion control algorithm MP-OL is proposed in Section IV. To validate the effectiveness of MP-OL, extensive performance evaluation is conducted in Section V. At last, conclusions are drawn in Section VI.

II. Related Work

A. Learning-Based Congestion Control

Machine learning is applied to congestion control to provide flexible strategies and cope with changing network conditions. Remy [13] is the first scheme to apply machine learning to congestion control. Remy takes the target network assumption as a prior knowledge, and generates a table that maps network states to the corresponding actions of congestion window (*cwnd*) under the guidance of an objective function. But the premise of ensuring Remy's performance is that the network assumption is not violated. Hence, improved schemes based on deep reinforcement learning (DRL) appear later (e.g., DRL-CC [7], Aurora [15], DeepCC [16] and DRL-TE [20]). These schemes have better general applicability because they all train an agent in advance and improve its behavior through

continuous interaction with the environment. Meanwhile, their disadvantages are equally notable. For example, the overall time complexity and space complexity of the schemes using the classical model of convolutional neural network (CNN) are:

$$\text{Time} \sim O\left(\sum_{l=1}^{D} M_l^2 \cdot K_l^2 \cdot C_{l-1} \cdot C_l\right), \quad (1)$$

$$\text{Space} \sim O\left(\sum_{l=1}^{D} K_l^2 \cdot C_{l-1} \cdot C_l + \sum_{l=1}^{D} M_l^2 \cdot C_l\right), \quad (2)$$

where D is the number of convolution layers of the CNN, l is the lth convolution layer, M_l^2 is the size of the output feature map in convolution layer l, K_l^2 is the size of the convolution kernel in convolution layer l, C_l is the number of convolution kernels in convolution layer l. The time complexity of the CNN is a superposition of the time complexities of all convolution layers, and the space complexity of the CNN is jointly determined by the total number of parameters and the sizes of the output feature map of layers. Therefore, these DRL-based schemes are prone to suffer from high complexity, resulting in a large amount of time and space consumed on model training and prediction. It also has more difficulties in quickly updating the model.

By contrast, schemes based on online learning (e.g., MPCC [9], Vivace [17] and PCC [21]) have no need to directly interpret the environment or utilize the previous experience, and thus do not generate high time and space complexity when training the model. These schemes usually have a longer convergence time but can react quickly to the changes of network conditions without updating the model. For example, MPCC [9] and Vivace [17] both design a utility function and apply it to convex optimization, and decide their sending rates through trial-and-error. Note that the design of utility function is very important and deserves careful consideration. Especially for multipath schemes, the design of utility function is more complicated than single path schemes, which should consider not only the global efficiency and fairness, but also what changes should be made when the utility function is applied to the subflow level.

B. Energy Issues and Data Issues of Machine Learning

As discussed in Section I, the development of ML models is mainly limited by the cost of R&D and practicality, especially for those who employ off-line learning. First, the accuracy improvement of ML models depends on the availability of huge computational resources, with the cost of electricity, hardware and compute time for training and development. Taking DRL for example, Strubell *et al.* [18] estimate the cost of cloud compute and electricity for training four important models [22], [23], [24], [25]. The results show that training a NAS [25] model with 213M parameters costs \$942,973∼\$3,201,722 USD on cloud compute, consumes about 656,347 kWh of electricity, and emits 626,155 lbs of CO_2, and higher associated costs are incurred when the dataset is updated or the model needs to be optimized. Second, we have more difficulties in the development of ML models during practical production. Due to the shortage of data resources, ML is moving towards privatization. According to a research [26], about 70% of companies need more than 100,000 labeled data items to ensure the confidence of production-level model, but encounter challenges with training data quality and quantity. For existing intelligent network control schemes, the source of training data has always been a controversial issue. Researchers generally generate training data on their own (e.g., using network simulator ns-3 [27]) since there are few public datasets available in this field. When the self-generated datasets deviate from reality, the effectiveness of the model will be severely affected. Therefore, due to the needs for huge efficient datasets and high training costs to ensure the reliability of an ML model, the deployability and practicality of ML-based schemes are severely restricted.

C. Environment Assumption for Multi-Armed Bandit

In the process of congestion control, it is very necessary to explore and exploit to find the optimal sending rate, and multi-armed bandit is a reference solution to handle the explore/exploit dilemma. Most stochastic bandit algorithms assume that they are carried out in a stationary environment. But the actual environment is not that ideal. To lift the restriction of stationary environment assumption and make bandit algorithms more suitable for real-world applications, there are also works studying the non-stationary bandit problems [28], [29], [30]. Piecewise stationary environment is a typical non-stationary environment setting. It divides the entire time period into S intervals, where S is a looser measure of nonstationarity, and the expected reward of each arm is assumed to be fixed in an interval but change between different intervals. Algorithms are designed for various bandit settings, with knowledge of interval S [31], [32] or unknown S [29].

III. Achieving Valid Congestion Control Through Online Learning

In this section, we discuss how to achieve valid congestion control from an online learning perspective. We model the multipath congestion control problem as a Multi-Armed Bandit (MAB) problem, and further simplify and improve this model to endow it with the ability to adapt to the actual networks with dynamic uncertainty. Through theoretical analysis, we prove the rationality of carrying out phased and independent online learning, and elaborate the basic methods for a decision-maker to learn the appropriate sending rate, so as to lay the groundwork for the detailed design of MP-OL in the next section.

A. Decision-Making Framework Construction

We model the general decision-making framework of congestion control as: a Decision-Maker (DM) interacting with a MAB system. DM selects an action per turn in order, and obtains a reward from the probability distribution related to this action. For end-to-end congestion control, the sender plays the role of DM, and every action it takes is a form of gambling with unknown results. The rewards can be reflected in throughput performance, delay, packet loss, etc.

From the view point of communication, highly dynamic networks belong to non-stationary environment but can be transformed into piecewise stationary environment. Assume that the regret of each arm is affected by a bandit parameter θ, and θ changes arbitrarily at arbitrary time, resulting in dynamic regret. Since it is hard to analyze or model dynamic regret in real network, we quantify the quality of algorithm by calculating the difference (i.e., regret) between the reward of the selected arm and the reward of the optimal arm.

The following problems make the decision-making of DM more difficult:

- The probability distribution of reward for each arm in MAB system is unknown to DM.
- The rewards observed by DM are incomplete. It can only observe the random reward of the selected arm.
- The reward distribution is not fixed, since the network is dynamic and new flows unceasingly join in the network, resulting in the change of network performance metrics.

To ensure the effectiveness of the decision-making framework and meet the demands of congestion control under uncertain network conditions, DM should consecutively interact (exploit and explore) with the environment to update the reward of each arm, so as to adapt to changes.

B. Problem Formulation

Let $\mathcal{A} = \{1, \ldots, A\}$ be the set of arms, and $t = 1, 2, \ldots,$ be the sequence of decision round. At each round *t*, the sender chooses one arm from $\mathcal{A}$ and obtains a regret of $\mathcal{R}_t^a$. For the convenience of calculations, we assume that the set of arms has only two elements, i.e., $\mathcal{A} = \{a_1, a_2\}$, a_1 means increasing the sending rate and a_2 means decreasing the rate. The specific variation quantity will not be discussed for the moment, but will be mentioned in the subsequent scheme design.

In the MAB problem, the regret of arms can be obtained by means of the regret mapping function calculation. Let $f_\theta : r_t = f_\theta(x_t)$ represents the reward mapping function governed by the environment's bandit parameter θ, which means that the learner chooses arm a_t at round *t* and gets the corresponding reward r_t. In this paper, we calculate the rewards based on the utility value of a connection. Thus, we have:

$$f_\theta : r_t = f_\theta(x_t) = U(x_t), \tag{3}$$

where $U(x_t)$ denotes the utility value of a flow (or a subflow for MPTCP connections), which means that the sender transmits data at rate x_t and obtains a utility of $U(x_t)$ in round *t*. It is worth mentioning that the utility function can be flexibly designed, and its form can be changed according to the influential degree of different parameters.

Before starting a new round, the sender needs to decide whether to increase or decrease the sending rate. But since it is unknown to the sender which arm can come with the highest reward, it needs to search for the appropriate sending rate through exploitation and exploration in the next few rounds.

Suppose the sender chooses arm a_1 in round $t + 1$ and obtains $U(x_{t+1})$, then the regret $\mathcal{R}_t^{a_1}$ can be calculated as:

$$\mathcal{R}_t^{a_1} = r_{t+1} - r_t = U(x_{t+1}) - U(x_t). \tag{4}$$

If we consider choosing a_1 as the exploitation process by default, then arm a_2 should be chosen in the next round for exploration. Thus, in round $t + 2$ the sender obtains $U(x_{t+2})$ and the corresponding regret $\mathcal{R}_t^{a_2}$ is:

$$\mathcal{R}_t^{a_2} = r_{t+2} - r_t = U(x_{t+2}) - U(x_t). \tag{5}$$

So far, we spend two rounds on determining the regret of each arm and the arm $a^* \in \mathcal{A}$ with the highest regret can be obtained. Thus, in round $t + 3$, the sender will choose arm a^* since it is more likely to have the highest expected regret. In the subsequent rounds, the sender repeats the above steps.

Remark 1 (Using MAB Rather Than Optimization Method): Many existing congestion control designs adopt the method of optimization. Optimization theory always assumes that external conditions are given, and other factors can be all controlled by one decision-maker (DM). Therefore, the optimization process is a deterministic process. However, such an assumption does not conform to the actual network conditions, as there are always multiple users involved in the network, who are not necessarily rational and sometimes have inconsistent preferences, and the behavior of each user may cause the change of the whole network conditions. For multipath transport, the traditional methods also assume that the relationship between different flows is competitive, while subflows cooperate to achieve possible highest profit. However, the competitive relationship between connections is difficult to be embodied in the design. Different from classical mathematical optimization, game theory is concerned with the study of multi-person decision problems, and the variables that affect the outcome are manipulated by multiple DMs, which clearly conforms better to the characteristics of actual network. Therefore, we do not adopt the traditional optimization methods (e.g., convex optimization) when designing our scheme, but design the utility function by considering the fact that subflows or flows interact with each other. Our design utilizes a method based on MAB to conduct exploration and exploitation.

Remark 2 (Suggestion on the Variation Quantity of Rate): In order to ensure that the sending rate can quickly converge to the optimal value, the variation quantity should be a variable that can be adjusted adaptively according to the network state rather than a constant. According to [17], we convert the gradient of the utility function into the change in rate to incrementally gain confidence in their decisions. As stated above, the regret of choosing each arm in round *t* has been given, we can determine the direction and quantity of rate variation according to $\mathcal{R}_t^{a_1}$ and $\mathcal{R}_t^{a_2}$. Let $Reg_t = \mathcal{R}_t^{a_1} - \mathcal{R}_t^{a_2}$, which represents the regret of choosing arm a_1 rather than arm a_2 in round *t*. If $Reg_t > 0$, the sender gains more confidence in increasing the rate, and vice versa. Let Δx be the difference between the rates of two arms, the gradient of the utility function is:

$$\gamma = \frac{Reg_t}{\Delta x} = \frac{\mathcal{R}_t^{a_1} - \mathcal{R}_t^{a_2}}{\Delta x} = \frac{U(x_{t+1}) - U(x_{t+2})}{x_{t+1} - x_{t+2}}, \tag{6}$$

which indicates the rate of return (RoR) of selecting arm a_1. The larger γ is, the more likely it is to get a high reward by increasing the sending rate. Otherwise, if γ is negative, consider reducing the rate. Let $|\gamma|$ be the absolute value of γ,

the greater $|\gamma|$ is, the greater the variation quantity of the rate is. Hence, in round $t+3$, the sender should transmit data at rate $x_{new} = x_t + \beta\gamma$, where $\beta > 0$ is a conversion factor.

Remark 3 (Abrupt Changes in the Environment): As suggested in [30], the bandit parameter θ shifts at unknown time instants, but remains constant in every epoch, i.e., the regret distribution between any two continuous change points remains unchanged, which can be represented as:

$$\underbrace{r_0, r_1, \ldots, r_{t_{c_1}-1}}_{distributed\ by\ f_{\theta_{c_0}}}, \ldots, \underbrace{r_{t_{c_S}}, r_{t_{c_S}+1}, \ldots, r_T}_{distributed\ by\ f_{\theta_{c_S}}}, \tag{7}$$

where $\{t_{c_i}\}_{i=1}^{S}$ is the set of the change points of regret distribution and $\{\theta_{c_i}\}_{i=0}^{S}$ is the corresponding set of bandit parameters. The sender has no knowledge of $\{t_{c_i}\}_{i=1}^{S}$ and $\{\theta_{c_i}\}_{i=0}^{S}$, and f_θ can be extended to complicated dependency structures without affecting the design of our scheme.

The scenario described by Eq. (7) fits the actual network in a certain way: The network state may remain stable over a period of time, but change significantly when certain events arrive (e.g., unexpected traffic burst and handoff). Bandit parameter θ is considered as the characteristic of environment, and different network states also correspond to different θ. These characteristics should be used as the contexts of online learning, and the online learning system produces different results according to different contexts. Therefore, the online learning processes under different network states can be independent of each other.

We decide to adopt a phased online learning approach that can respond to the variation of network state. When the characteristic of the environment changes, the online learning process will enter a new phase, and no longer care about the information of the previous phase.

IV. Online Multipath Congestion Control Design

In this section, we present the detailed design of MP-OL. The procedure of MP-OL includes slow start, online learning and network state monitoring. We design a utility function as the optimization objective of online learning, which considers the cooperative relations between subflows or flows and enables flows to achieve a win-win result through cooperative game. As discussed in Section III-B, MP-OL adopts phased online learning. After finishing slow start, MP-OL starts online learning and monitors the dynamic change of network through a lightweight clustering method. When there is an obvious change in network state, MP-OL will start a new learning process. In general, MP-OL is practical and easy to deploy since it only requires sender-side changes and does not need datasets or pre-trained models. The basic framework of MP-OL algorithm is illustrated in Fig. 1.

A. Per-Subflow Utility Function of MP-OL

Before designing the utility function, we first consider a scenario in which m participants (flows/users) compete in the same environment. At this point, the ideal situation is that they are engaged in cooperative game. We hope that their respective payoffs can converge to similar values, so as to ensure

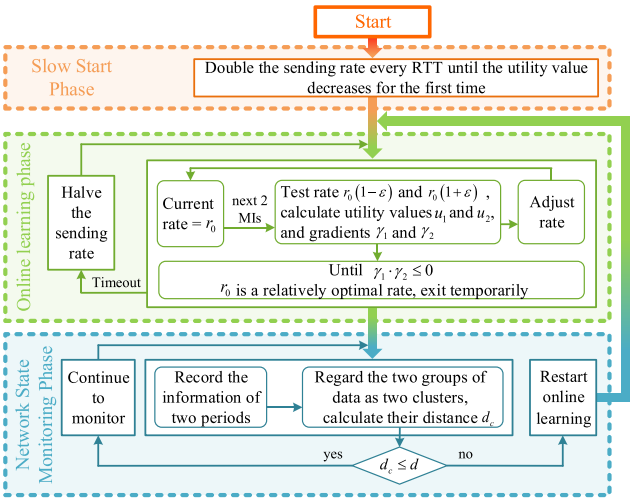


Fig. 1. Algorithm framework of MP-OL.

intra-protocol fairness. Meanwhile, if some actions of one participant deviate from the prescribed strategy, the other participants will show small regret to ensure that their sequences of actions do not change radically. Similarly, for an MPTCP connection with multiple subflows, all its subflows are also engaged in cooperative game, and such relationships should be taken into account when designing the utility function.

Consider an MPTCP connection f consists of a set of subflows d_f. For one subflow $j \in d_f$, we design its utility function as:

$$U_j = \left(\sum_{i\in d_f, i\neq j} x_i\right)^a + {x_j}^a - b x_j L_j - c x_j \frac{d(RTT_j)}{dT}, \tag{8}$$

where $\sum_{i\in d_f, i\neq j} x_i$ denotes the aggregate of sending rates of all subflows belonging to connection f except for subflow j. L_j, x_j and $\frac{d(RTT_j)}{dT}$ denote the loss rate, sending rate and latency gradient [17] of subflow j, respectively. a, b and c are constants with the value range of $0 \le a < 1$, $b > 0$, $c \ge 0$.

Eq. (8) includes two parts: *connection part* and *subflow part*. The *connection part* uses the sending rates of other subflows to evaluate the impact of the actions of subflow j on the whole connection f. While the *subflow part* only cares about the impact of subflow j on itself.

Remark 4 (Loosely Coupled Subflows): In Eq. (8), subflows are designed to be loosely coupled with each other. This is because when MPTCP is used in practice, its subflows usually go through different networks, and the loss rate and latency gradient experienced by each subflow are not necessarily coincident. Thus, each subflow should have higher independence and mainly learn the information of its own link, so as to avoid being affected by subflows on the worse links.

Remark 5 (Cooperative Game and Convergence to a Fair Equilibrium): The utility function is designed to encourage all flows involved to play the game in a cooperative manner.

When subflow j shares the same bottleneck link with other $m-1$ subflows, even if subflow j increases its sending rate and takes up more bandwidth than its fair share, the other

subflows will reduce their rates accordingly. As a result, subflow *j* receives a negative feedback from the *connection part*, which stimulates it to adjust its sending rate to the right direction in the next couple of rounds. Therefore, the sending rates of *m* subflows will eventually converge to an equilibrium point $(x_1^*, x_2^*, \ldots, x_m^*)$ with $x_1^* = x_2^* = \cdots = x_m^*$.

Next, considering more general cases, regardless of whether some subflows are sharing the same bottleneck link. Use d_l to represent the set of flows that pass link *l*, where $j \in d_l$.

In Eq. (8), if the impact of other flows (*j* excluded) in link *l* is not considered, then L_j and $\frac{d(RTT_j)}{dT}$ can be regarded as constants, and we have:

$$\frac{\partial U_j}{\partial x_j} = a{x_j}^{a-1} - bL_j - c\frac{d\left(RTT_j\right)}{dT}, \tag{9}$$

$$\frac{\partial^2 U_j}{\partial {x_j}^2} = a(a-1){x_j}^{a-2} < 0. \tag{10}$$

Therefore, Nash equilibrium [33], [34] exists. Set X_1^j as the Nash equilibrium, where $\frac{\partial U_j(X_1^j)}{\partial x_j} = 0$, $X_1^j > 0$. Considering the fact that the actions of the participants are interacted, we assume the relationships between loss rate, latency gradient and the aggregate sending rate of link *l* as:

$$L_j = 1 - e^{-\frac{\sum_{i\in d_l} x_i}{B}}, \tag{11}$$

$$\frac{d\left(RTT_j\right)}{dT} = T - e^{-\frac{\sum_{i\in d_l} x_i + R_2}{R_1}}, \tag{12}$$

where *B*, *T*, R_1 and R_2 are constants with the value range of $B, R_1 > 0$, $T \geq 1$, $R_2 \leq 0$.

First, suppose all flows are still engaged in non-cooperative game. So there are:

$$\frac{\partial U_j}{\partial x_j} = a{x_j}^{a-1} + \left(b - \frac{bx_j}{B}\right)e^{-\frac{\sum_{i\in d_l} x_i}{B}} + \left(c - \frac{cx_j}{R_1}\right)e^{-\frac{\sum_{i\in d_l} x_i + R_2}{R_1}} - b - cT, \tag{13}$$

$$\frac{\partial^2 U_j}{\partial {x_j}^2} = a(a-1){x_j}^{a-2} + \left(\frac{bx_j}{B^2} - \frac{2b}{B}\right)e^{-\frac{\sum_{i\in d_l} x_i}{B}} + \left(\frac{cx_j}{{R_1}^2} - \frac{2c}{R_1}\right)e^{-\frac{\sum_{i\in d_l} x_i + R_2}{R_1}}, \tag{14}$$

where exists $X_2^j > 0$ such that $\frac{\partial U_j(X_2^j)}{\partial x_j} = 0$, and X_2^j is negatively correlated with $\sum_{i\in d_l, i\neq j} x_i$. Because each flow only cares about individual rationality and tends to increase its sending rate, while does not want any other flows to increase their sending rate, the final Nash equilibrium satisfies: $\forall i, k \in d_l, X_2^i = X_2^k$.

Second, suppose all flows are engaged in cooperative game. Take all flows in set d_l as an example, and d_l is a grand coalition consists of *q* members. Set $\mathrm{X} = \sum_{i\in d_l} x_i$, so the utility function of the coalition is:

$$U_l(\mathrm{X}) = \mathrm{X}^a - b\mathrm{X}\left(1 - e^{-\frac{\mathrm{X}}{B}}\right) - c\mathrm{X}\left(T - e^{-\frac{\mathrm{X}+R_2}{R_1}}\right), \tag{15}$$

and we have:

$$\frac{\partial U_l}{\partial \mathrm{X}} = a\mathrm{X}^{a-1} + \left(b - \frac{b\mathrm{X}}{B}\right)e^{-\frac{\mathrm{X}}{B}} + \left(c - \frac{c\mathrm{X}}{R_1}\right)e^{-\frac{\mathrm{X}+R_2}{R_1}} - b - cT, \tag{16}$$

$$\frac{\partial^2 U_l}{\partial \mathrm{X}^2} < 0. \tag{17}$$

There exists $X_3 > 0$ such that $\frac{\partial U_l(X_3)}{\partial \mathrm{X}} = 0$, which is the Nash equilibrium of the game between coalition and "Nature". And it satisfies: $\forall i \in d_l, X_3/q \geq X_2^i$, so there is $U_l(X_3) \geq \max \sum_{i\in d_l} U_j(x_i)$, which means the coalition can create most cooperative surplus and acquire higher utility than non-cooperation when all flows $i \in d_l$ are engaged in cooperation, so no participant is willing to withdraw from the grand coalition and make independent decisions.

After the grand coalition obtains the effective aggregate sending rate $X_3 = \sum_{i\in d_l} x_i$, the next step is to allocate the sending rate fairly within the coalition. Suppose d_l contains *q* flows and all participants are of equal status, the equilibrium point $(x_1^*, x_2^*, \ldots, x_q^*)$ of final allocation result will satisfy $x_1^* = x_2^* = \cdots = x_q^* = X_3/q$.

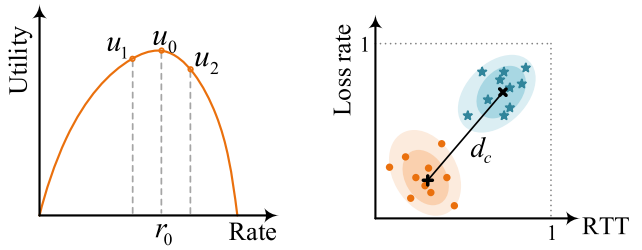
Therefore, compared with non-cooperative game, flows can achieve higher sending rates and utility values when we emphasize both individual rationality and collective rationality. The above steps prove that flows on link *l* can obtain effective and fair sending rate through cooperative game. Similarly, this conclusion can also be extended to the flows in a whole network.

B. Per-Subflow Rate Control of MP-OL

As depicted in Fig. 1, MP-OL consists of three phases and each of which adopts different rate control methods. MP-OL starts with Slow Start Phase, and then iteratively executes Online Learning Phase and Network State Monitoring Phase multiple times. The iteration continues until the transfer completes or the connection terminates. Each subflow optimizes its rate independently and asynchronously, and shifts between these three phases without affecting other subflows. We divide time into consecutive Monitor Intervals (MIs). Each MI monopolizes one transmission round, which means that the duration of one MI is 1 *RTT* (Round-Trip Time). For any subflow $j \in d_f$, it selects a sending rate at the beginning of each MI, and calculates its utility value according to Eq. (8) at the end of the MI.

1) Slow Start Phase: MP-OL starts at Slow Start Phase. Subflow *j* selects a sending rate at the beginning of the first MI, and then doubles its sending rate every *RTT* until the utility value decreases for the first time. Then, it exits Slow Start Phase permanently.

2) Online Learning Phase: After exiting Slow Start Phase, MP-OL enters Online Learning Phase. As stated in Section III-B, MP-OL will spend two MIs deciding whether to increase or decrease the sending rate. Suppose the current sending rate of subflow *j* is r_0, and the utility value is u_0. In the next two MIs, the sender adopts the sending rate of $r_0(1-\varepsilon)$ and $r_0(1+\varepsilon)$ respectively and gains the corresponding utility values u_1 and u_2, where ε decides the *learning step* of senders. In the process of exploitation and exploration, we only plan to learn the direction to which the rate should be changed. Thus, we choose small *learning step* to contribute



(a) Controlling rate with concave utility function

(b) Two clusters and the distance between them

Fig. 2. Illustrations of the methods used in Online Learning Phase and Network State Monitoring Phase.

to the stability of the algorithm. The range of ε is recommended as $\varepsilon \in [0.1, 0.01]$, and in the experimentation, we choose $\varepsilon = 0.05$. So far, the sender has selected arm a_1 and a_2 successively, and it needs to calculate two gradients of the utility function to estimate the best arm a^*:

$$\gamma_1 = \frac{u_0 - u_1}{\varepsilon r_0}, \quad \gamma_2 = \frac{u_2 - u_0}{\varepsilon r_0}. \tag{18}$$

The family of utility functions in Eq. (8) belongs to the category of "socially-concave" when $a \leq 1$ in game theory [35]. We depict them as Fig. 2(a). According to the characteristic of the utility function: If $\gamma_1 \cdot \gamma_2 > 0$, and γ_1 and γ_2 are both positive, increase the sending rate in the next MI. If γ_1 and γ_2 are both negative, decrease the sending rate in the next MI. According to Eq. (6), the variation quantity of sending rate Δr changes dynamically and can be calculated as:

$$\Delta r = \beta \frac{u_1 - u_2}{r_0(1-\varepsilon) - r_0(1+\varepsilon)} = \beta \frac{u_2 - u_1}{2\varepsilon r_0}, \tag{19}$$

so the sending rate adopted in the next MI is $r_{new} = r_0 + \Delta r$.

From Fig. 2(a) we can find that when the sending rate is close to the optimal value, the effect of using the gradients of utility function to adjust the sending rate will be no longer satisfactory. If we insist on using online learning, the algorithm will fall into a dilemma of slow convergence but high computing overhead. Therefore, MP-OL exits Online Learning Phase temporarily when $\gamma_1 \cdot \gamma_2 \leq 0$ and adopts a low-complexity traditional method to fine tune its congestion windows (*cwnd*).

3) Network State Monitoring Phase: In Network State Monitoring Phase, *cwnd* retains the values learned in the previous phase, and MP-OL adopts a coupled window adjustment strategy based on additive increase multiplicative decrease (AIMD) to continue to fine-tune *cwnd*, and the detailed adjustment method is the same as LIA [3]. Meanwhile, the sender also regularly monitors the change of network state. If it is detected that the network state has changed significantly, the sending rate learned previously will be no longer appropriate. Thus, subflow *j* needs to re-enter Online Learning Phase to learn a new sending rate suitable for current network conditions. Otherwise, subflow *j* continues monitoring.

MP-OL applies the method of clustering analysis to monitor the variation of network state. We set *n* MIs as one Monitoring Period (MP), during which the sender will record the loss rate *l* and latency *r* experienced by subflow *j* in each MI, represented as $P = \{(l_1, r_1), (l_2, r_2), \ldots, (l_n, r_n)\}$. Then the sender accesses the information of the previous MP: $P' = \{(l_{n+1}, r_{n+1}), (l_{n+2}, r_{n+2}), \ldots, (l_{2n}, r_{2n})\}$. It regards these two sets of data as two clusters and calculates the distance d_c between them. In Network State Monitoring Phase, MP-OL only records the network state information of two MPs: the current MP and the previous MP. There is no need to store any more history information.

Before clustering analysis, we should first normalize these two sets of data. MP-OL adopts Min-Max Normalization to perform a linear transformation on the original data, and make them fall in the range of [0, 1], so as to transform loss rate and latency into dimensionless quantities, while preserving the relationships among the original data values. The calculation method is given below:

- For sequence $l_1, l_2, \ldots, l_n, l_{n+1}, l_{n+2}, \ldots, l_{2n}$, perform:

$$l_i^* = \frac{l_i - \min\limits_{1 \leq j \leq 2n} \{l_j\}}{\max\limits_{1 \leq j \leq 2n} \{l_j\} - \min\limits_{1 \leq j \leq 2n} \{l_j\}}, \tag{20}$$

such that a new sequence $l_1^*, l_2^*, \ldots, l_n^*, l_{n+1}^*, l_{n+2}^*, \ldots, l_{2n}^*$ is obtained, which is dimensionless and satisfies $l_i^* \in [0, 1]$.

- Similarly, for $r_1, r_2, \ldots, r_n, r_{n+1}, r_{n+2}, \ldots, r_{2n}$, the new sequence is $r_1^*, r_2^*, \ldots, r_n^*, r_{n+1}^*, r_{n+2}^*, \ldots, r_{2n}^*$.

So far, we have mapped the two original sets P and P' to two new sets $N = \{(l_1^*, r_1^*), (l_2^*, r_2^*), \ldots, (l_n^*, r_n^*)\}$ and $N' = \{(l_{n+1}^*, r_{n+1}^*), (l_{n+2}^*, r_{n+2}^*), \ldots, (l_{2n}^*, r_{2n}^*)\}$. As shown in Fig. 2(b), MP-OL regards set N and set N' as two clusters and calculates their respective centroids (L, R) and (L', R'), where $L = (l_1^* + l_2^* + \cdots + l_n^*)/n$, $R = (r_1^* + r_2^* + \cdots + r_n^*)/n$, and so is the calculation of L' and R'. Thus, the Euclidean distance d_c between these two clusters is:

$$d_c = \sqrt{(L - L')^2 + (R - R')^2}. \tag{21}$$

MP-OL sets a threshold d for the distance in advance. If $d_c \leq d$, the network state has not changed significantly, subflow *j* stays in Network State Monitoring Phase and continues monitoring. If $d_c > d$, the network state has changed significantly, therefore, subflow *j* re-enters Online Learning Phase to learn a new proper sending rate. Different combinations of *n* and *d* have different effects, we will test and analyse the values of *n* and *d* in the next section.

Remark 6 (A Discussion on the Complexity of MP-OL): Eq. (8) implies $O(n)$ time complexity (*n* is the number of subflows) of Online Learning Phase, which means MP-OL has the same time complexity as the traditional AIMD-based methods. The number of subflows is the main factor affecting the efficiency of MP-OL. When there is a large number of subflows, the $O(n)$ time complexity will lead to a decrease in efficiency. But given that MPTCP generally does not deploy too many subflows in practice, the effect of $O(n)$ time complexity on MP-OL can be neglected. At the same time, the space complexity of MP-OL is $O(1)$, for it only temporarily occupies a fixed size of storage space during operation. Therefore, MP-OL is superior to those schemes that employ off-line learning or DRL in terms of both time complexity and space complexity.

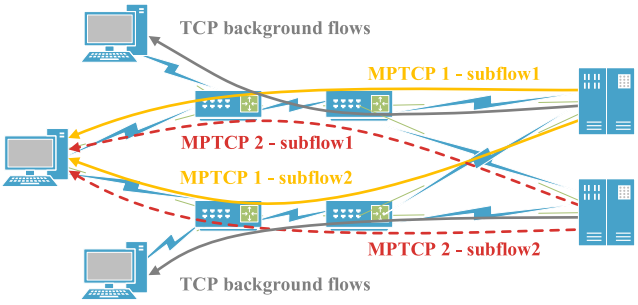


Fig. 3. The structure of platform used in the experiments.

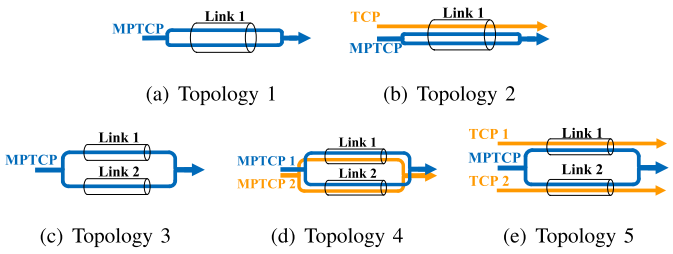


Fig. 4. Topologies that can be implemented in our laboratory platform.

Remark 7 (The Impact of the Environment Assumption on MP-OL): The magnitude of environment change determines whether our algorithm should react to the change. When it is small, re-using the previous model only comes with a very small additional regret. Thus, maintaining the previous learning model will not bring obvious negative influence, and might be more efficient and lower-cost than learning a new model.

V. Performance Evaluation

We implement the proposed algorithm MP-OL in the Linux kernel, and conduct sufficient experiments on our laboratory platform as well as in a real network environment. We choose Olia [4], Balia [36], wVegas [37] and MPCC [9] as the contrastive MPTCP variants, where Olia, Balia and wVegas are all implemented in the Linux kernel, and MPCC is an online-learning-based MPTCP congestion control algorithm proposed recently. In this section, as recommended in [17], we set $a = 0.9$, $b = 11$, $c = 1$. Unless otherwise specified, MP-OL sets $n = 5$ and $d = 0.5$ by default. The reason for this choice will be introduced in subsequent experiments. Fig. 3 depicts the structure of our deployed laboratory platform. All the servers and clients are running on Linux ubuntu 16.04 OS with MPTCP kernel version 4.19.196 [38]. In this platform, we can implement the topologies shown in Fig. 4 to simulate different network scenarios for testing.

Note that the application of MP-OL is not limited to MPTCP, its congestion control method is universal among various multipath protocols. For example, the congestion control methods of MPTCP and MPQUIC [39] are basically the same, but considering that MPQUIC traffic only accounts for a very small part of all UDP traffic at present, and still lacks technical achievements that turn research into revenue. Therefore, we implement MP-OL based on MPTCP to provide more adequate performance comparison.

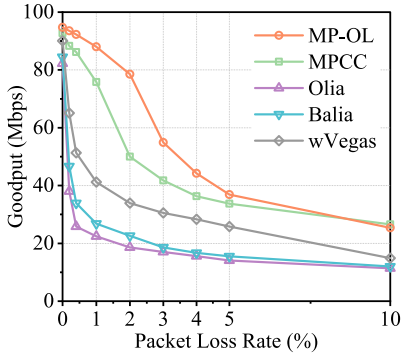


Fig. 5. Goodputs of MPTCP variants under varying random packet loss rates.

A. Resilience to Non-Congestion Loss

We first test the resilience to non-congestion loss of our algorithm. In real networks, packet loss is not only caused by congestion, but also by interference, link failure and handover. The design of MP-OL includes some anti-packet loss measures. To validate this feature, we employ the topology of Fig. 4(a), and set the delay to 30*ms* and bandwidth to 100*Mbps*. We add different random packet loss rates on *Link*1, run iperf3 [40] for 120 seconds, and count the goodput of each MPTCP variant. The result is illustrated in Fig. 5.

In Fig. 5, as the random packet loss rate increases from 0% to 10%, the goodputs of the variants of MPTCP decrease accordingly, but at different rates, which means that the degree to which loss rate influences goodput varies with the algorithm. As loss-based congestion control algorithms, Olia and Balia both perform poorly in non-congestion loss environment even if the loss rate is as low as 0.2%. When the loss rate reaches to 3%, their goodputs decrease by more than 75%. Delay-based algorithm wVegas is also unable to achieve high goodput in the whole random loss rate range. MPCC can guarantee higher goodput when loss rate does not exceed 1%, but its goodput still drops rapidly to less than half its best value at any higher loss rate. This means that the random packet loss rate that MPCC can tolerate is less than 2%. MP-OL has better performance over a wider range. The goodput curve of MP-OL decreases the slowest among all the curves, which means that MP-OL maintains higher goodput across the entire range and can tolerate higher random packet loss rate than all other MPTCP variants. Even if there are a lot of random lost packets, MP-OL can try higher sending rates since it relies on the gradient of the utility function and adjusts its sending rate based on exploitation and exploration. Therefore, MP-OL has more chances to achieve a higher goodput under high random packet loss rates. Specifically, the performance benefit of MP-OL is significant when loss rate is in the range of $0\% \sim 2\%$, and as loss rate increases to 3%, MP-OL can still achieve 55% of its best goodput, which significantly outperforms MPCC.

B. TCP-Friendliness and Intra-Protocol Fairness

In this part, we use the topology of Fig. 4(b) to study the TCP-friendliness, and investigate how different MPTCP variants interact with delay-based TCP (Vegas [41]) and loss-based TCP (Cubic [42]). The delay and bandwidth of *Link*1 are set to 30*ms* and 100*Mbps*, respectively. And we use the

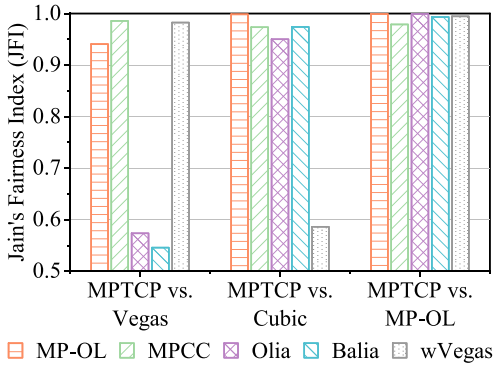


Fig. 6. Jain's fairness index of different MPTCP variants.

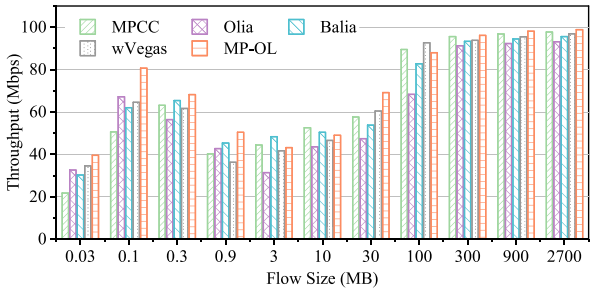


Fig. 7. File download performance of MP-OL and other four MPTCP variants.

topology of Fig. 4(d) to study the intra-protocol fairness and test how MP-OL interacts with different MPTCP variants, in which the delay and bandwidth of *Link*1 and *Link*2 are both set to 30*ms* and 50*Mbps*, respectively. We choose Jain's fairness index (JFI) [43], a widely known metric used for evaluating the fairness of resource allocation among traffic flows, to intuitively compare the fairness of different MPTCP variants. We run iperf3 for 120 seconds and calculate the JFI for different MPTCP variants in three cases. The values of JFI lie in the range of [0.5, 1], where the value of 1 indicates maximal fairness. The results are shown in Fig. 6.

First, we let MPTCP subflows share the same bottleneck link with a TCP Vegas flow. Experimental results show that Olia and Balia obtain JFIs close to 0.5, which means they monopolize almost all the bandwidth resources and are very unfriendly to delay-based TCP. The JFIs of MPCC and wVegas are very close to 1, which means they can share the bandwidth with delay-based TCP with almost perfect fairness. MP-OL's JFI is slightly lower than that of MPCC or wVegas because it adjusts its *cwnd* according to the traditional method in Network State Monitoring Phase (i.e., *cwnd* is increased by 1 for each RTT). But when in cooperation with Online Learning Phase, to delay-based TCP, MP-OL still shows significantly better friendliness than Olia or Balia. Second, we let MPTCP subflows share the same bottleneck link with a TCP Cubic flow. The results show that the JFI obtained by MP-OL is extremely close to 1 and significantly better than loss-based MPCC, Olia and Balia. While wVegas can hardly compete with loss-based TCP, achieving the worst system fairness, most of its bandwidth is preempted by Cubic. MP-OL provides the best friendliness to loss-based TCP among the five MPTCP variants. Finally we test the intra-protocol fairness and the friendliness of MP-OL to different types of MPTCP variants. The MPTCP variants involved can be divided into three types: online-learning-based (MP-OL, MPCC), loss-based (Olia, Balia) and delay-based (wVegas). The results show that when an MP-OL flow competes with another MP-OL flow, they can obtain a JFI extremely close to 1 and share the network resources equally, indicating that the intra-protocol fairness is satisfied. Moreover, no matter competing with loss-based, delay-based or online-learning-based MPTCP, MP-OL can always obtain JFIs that are close to 1 and ensure fairness.

On the whole, MP-OL has good TCP-friendliness, MPTCP-friendliness and intra-protocol fairness.

C. Transmission Performances for Different Sizes of Flows

The actual network is filled with various sizes and types of data, which is also an important factor affecting the performance of MPTCP. According to the sizes of data flows, the existing researches usually classify flows into two categories: mice flows and elephant flows. For different categories of flows, the key factors that affect the transmission performance are different. For mice flows, the key factors could be the design of slow start phase or delay. While for elephant flows, it could be bandwidth.

This section studies the ability of MPTCP variants to deal with different categories of flows. As for the classification of flows, we refer to the definition in [44]: Flows with sizes less than 1*MB* are mice flows, while flows with sizes larger than 1*MB* are elephant flows. We adopt the topology of Fig. 4(c), and set the delay and bandwidth of *Link*1 and *Link*2 both to 30*ms* and 50*Mbps* respectively. We control the size of a flow by setting a file of a given size and using it for transmission. The larger the file size, the larger the flow size. The average throughputs for file downloads are illustrated in Fig. 7.

We observe that MPCC performs very poorly in transmitting mice flows, especially when the flow size is less than 0.3*MB*. When transmitting 0.03*MB* files, MPCC can only achieve about 50% of the throughput of Olia, Balia and wVegas, or 90% of the throughput of MP-OL. When the file size grows to 0.1*MB*, the throughput of MPCC begins to reach that of Olia, Balia and wVegas, but is still far inferior to MP-OL. We believe that the problem is in the slow start phase. MP-OL has the same exit condition for slow start as MPCC. They both exit slow start phase permanently once the utility value declines, but MP-OL still performs much better in mice flow transmission. We attribute it to the poor design of utility functions. MPCC only empirically expands the design of single path scheme PCC [17], [21] and uses the aggregate sending rate of all subflows in its utility function. As a result, the action of a single subflow is susceptible to the overall performance of the flow or the dynamic state variety of the network. Thus, the exit condition for slow start phase is more likely to be triggered accidentally, and thus quitting slow start untimely and entering the probing state with slow convergence speed.

With the increase of flow size, the defects of MPCC's slow start phase are gradually diluted. In contrast, MP-OL always

TABLE I
THE VARIATION OF NETWORK PARAMETERS IN SITUATION 1

Timeline (s)	0∼30	31∼60	61∼90	91∼120	121∼150
Link 1					
Delay (ms)	20∼40	150∼250	30	200	80∼120
Loss rate	0	2%	2%	0	0.5%
Link 2					
Delay (ms)	150∼250	80∼120	200	30	20∼40
Loss rate	2%	0.5%	0	2%	0

TABLE II
THE VARIATION OF NETWORK PARAMETERS IN SITUATION 2

Timeline (s)	0∼30	31∼60	61∼90	91∼120	121∼150
Link 1					
Delay (ms)	0∼40	10	50∼100	0∼200	0∼400
Loss rate	0∼10%	0∼5%	0∼2%	2%	0
Link 2					
Delay (ms)	0∼400	0∼200	50∼100	10	0∼40
Loss rate	0	2%	0∼2%	0∼5%	0∼10%

performs well in dealing with different categories of flows, and achieves the best overall performance. MP-OL's advantages is especially noticeable when transmitting mice flows. It achieves $25\% \sim 90\%$ higher throughput than MPCC, which is also $1\% \sim 40\%$ higher than wVegas, Olia and Balia. This is because we improve the design of the utility function of MP-OL such that a single subflow is less susceptible to the performance of other subflows or the dynamic state variety of the network, and can fully play the advantages of slow start phase to accelerate mice flows. As flow size increases from 10*MB* to 100*MB*, MP-OL reaches high throughput rapidly, which is basically consistent with MPCC, and it can guarantee the highest or near highest throughput, which is by no means inferior to other MPTCP variants. Due to the fixed network parameters and overall stable network state, MP-OL cannot give full play to its ability to optimize throughput under unstable network conditions, and hence does not show an obvious advantage when transmitting flows larger than 10*MB*. Anyway, MP-OL inherits the advantage on throughput performance of MPCC in transmitting elephant flows, and in addition achieves the performance improvement of mice flows. This part also proves the effectiveness of our hypothesis about the relationship between subflows or flows, and their cooperative relations are reflected in the design of our utility function, which is also proven to be effective.

D. Coping With Varying Network Configuration

Previous experiments are conducted under fixed network parameters. Next, we intend to study the impact of changing network parameters. We adopt the topology similar to Fig. 4(e), and run iperf3 for 150 seconds, during which we modify the network configuration regularly. In each round of testing, we set the bandwidths of *Link* 1 and *Link* 2 both to 200*Mbps* and change the delay and loss rate of each link, and delay jitter is also considered. We simulate the moderate variation and drastic variation of network state, which are called Situation 1 and Situation 2, respectively. Each MPTCP flow contains two subflows: *Subflow* 1 on *Link* 1 and *Subflow* 2 on *Link* 2. The variation of network parameters of *Link* 1 and *Link* 2 are shown in Table I and Table II.

We deploy the configuration of Situation 1 and Situation 2, respectively, and test the performance of MP-OL, MPCC, wVegas and Olia. The throughput curves along with time of each subflow are shown in Fig. 8 and Fig. 9. As seen in Fig. 8, MP-OL has better average throughput than the other MPTCP variants, and can accurately identify every change point of network state and make timely adjustments. Observing the characteristics of MP-OL curves, we can find that it

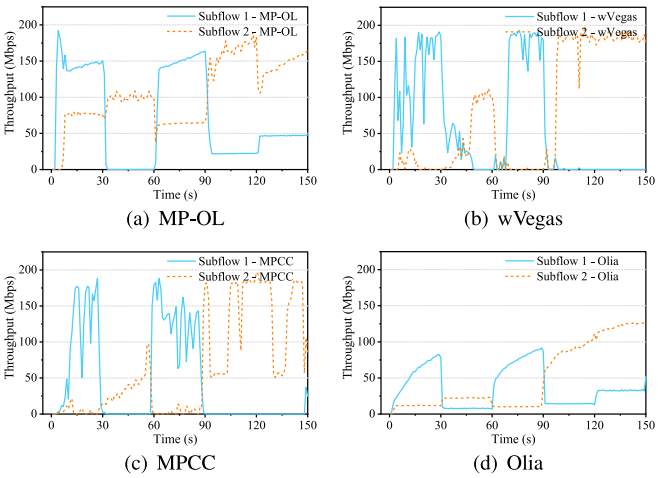
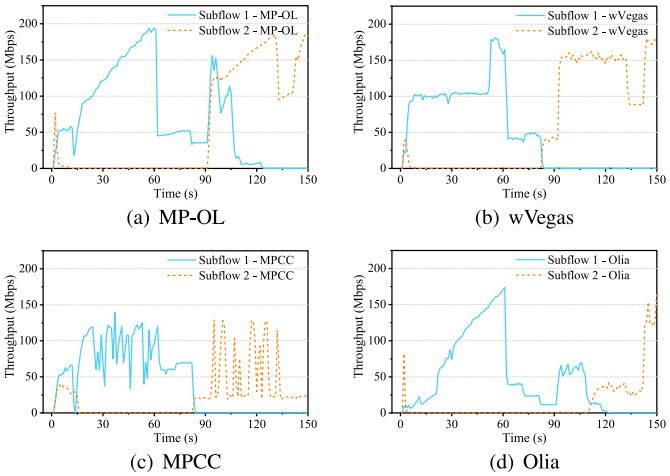


Fig. 8. Throughput comparison of MPTCP variants in Situation 1.

combines the advantages of MPCC and Olia, and achieves the highest bandwidth utilization among all MPTCP variants: First, the throughput of MP-OL converges to a relatively good value quickly, and then continues moving towards the optimal value. Second, MP-OL can always maintain a relatively stable throughput, while the throughput of MPCC and wVegas both fluctuate much more significantly. For some applications such as multimedia, the highly fluctuating throughput is undesirable [45], and our scheme MP-OL is thus more preferable. In Fig. 9 we reach similar conclusions. In addition, we find that the throughput of MP-OL can still gradually converge to the optimal value even when the network state changes drastically, while the throughputs of MPCC and wVegas tend to remain stagnant after reaching a low performance, and cannot carry on further exploration and improvement.

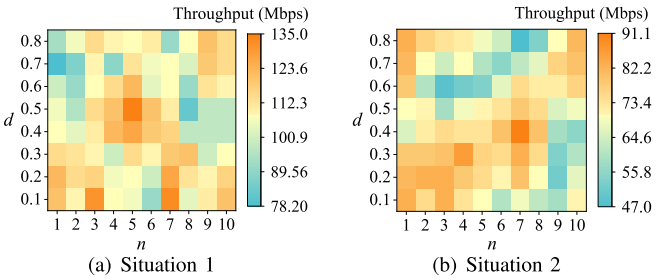
E. Performance Comparison Under Different Monitoring Parameters

In Section IV-B3, we introduce two monitoring parameters: Monitoring Period n and the threshold d for the distance between clusters. In our design, n determines the length of MP-OL's decision cycle and the impact of outliers on the decision results, and d determines the perception of MP-OL towards the change extent of network state. In the previous experiments, we set $n = 5$ and $d = 0.5$ by default, now we want to learn more about the influence of parameters n and d on the performance of MP-OL. We adopt the same topology, network configurations and operations as Section V-D, and set $n \in \{1, 2, \ldots, 10\}$ and $d \in \{0.1, 0.2, \ldots, 0.8\}$, which



(a) MP-OL (b) wVegas (c) MPCC (d) Olia

Fig. 9. Throughput comparison of MPTCP variants in Situation 2.

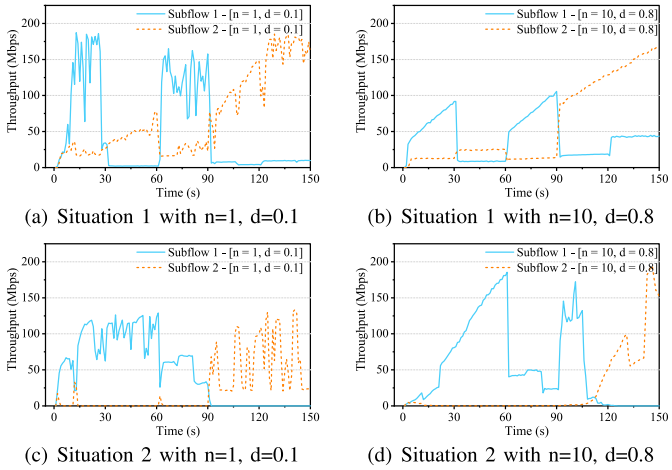


(a) Situation 1 (b) Situation 2

Fig. 10. The average throughput of MP-OL when selecting each combination of n and d in Situation 1 and Situation 2.

amounts to $10 \times 8 = 80$ different combination plans. We calculate the average throughput of MP-OL in each combination $[n, d]$. The results are illustrated in Fig. 10(a) and Fig. 10(b). We also plot the throughput curves of *Subflow*1 and *Subflow*2 along with time when parameters n and d reach their minima and maxima, namely $[n = 1, d = 0.1]$ and $[n = 10, d = 0.8]$. The results are shown in Fig. 11.

Fig. 10(a) and Fig. 10(b) can provide reference basis for the values of n and d. We infer that, in most cases, the combinations of a large n with a small d or a small n with a large d are not recommended, for they interfere with the model's judgment of network states. The recommended ranges of values for n and d are $n \in [4, 7]$ and $d \in [0.3, 0.6)$, but can also be modified according to actual conditions. Compare Fig. 11(a) with Fig. 8(c), Fig. 11(b) with Fig. 8(d), Fig. 11(c) with Fig. 9(c), and Fig. 11(d) with Fig. 9(d), we can further recognize the influence of parameters n and d on MP-OL's behavior. No matter how the network configuration changes, when we set $[n = 1, d = 0.1]$, the characteristics of the throughput curves of MP-OL share many similarities with MPCC. And when we set $[n = 10, d = 0.8]$, MP-OL behaves like Olia. This is because when $n = 1$, $d = 0.1$, MP-OL exits Network State Monitoring Phase easily and quickly, which means its behavior is almost completely controlled by Online Learning Phase. Similarly, when $n = 10$, $d = 0.8$, the cycle of Network State Monitoring Phase becomes longer and the exit condition is more difficult to be met, which means that MP-OL will spend the majority of time on Network State Monitoring Phase during the whole



(a) Situation 1 with n=1, d=0.1 (b) Situation 1 with n=10, d=0.8 (c) Situation 2 with n=1, d=0.1 (d) Situation 2 with n=10, d=0.8

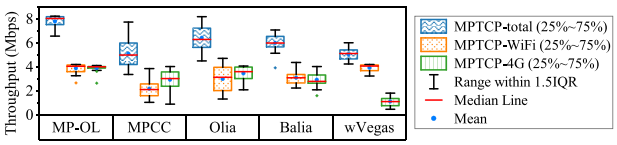
Fig. 11. The throughput curves of MP-OL when it sets $[n = 1, d = 0.1]$ and $[n = 10, d = 0.8]$ respectively in Situation 1 and Situation 2.

transmission process, and consequently degrades into an Olia experience.

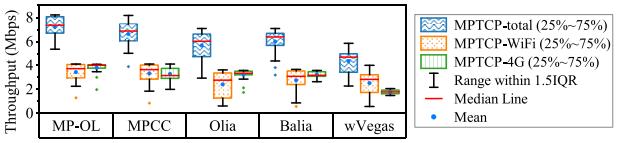
As to Monitoring Period n, the larger n is, the more time MP-OL will spend on monitoring, but the less easily the monitoring results are affected by outliers, resulting in good performance of the system in steady state. While with a smaller n, although MP-OL can respond more quickly to the variation of network state, it is more likely to suffer from inaccurate judgment and thus cause unstable state of the system. When n is very small, even if the network fluctuates slightly at a certain time, resulting in short-term loss rate fluctuation and latency fluctuation, MP-OL will misjudge some jitters within the normal range as significant changes of the network state. In conclusion, parameter n or parameter d does not exist independently, they collaborate to determine how MP-OL's Online Learning Phase and Network State Monitoring Phase fit together. Reasonable values are conducive to fully play the respective advantages of these two phases and achieve better performance through effective cooperation.

F. Benefits in Real Network

In addition to experiments on controlled networks, we also test our scheme in a real network environment. We deploy an MPTCP server on the cloud located in a foreign country and download a 100*MB* file from the cloud-based server using both Wi-Fi and 4G. Considering the diurnal patterns for Internet usage [46], we conduct our tests in two different periods of the day. The total throughput of MPTCP and the throughput of each subflow are shown in Fig. 12. It can be observed that wVegas is hard to work in real network, as it achieves the lowest total throughput and 4G only accounts for $20\% \sim 42\%$ of the total throughput. MPCC, Olia and Balia have unstable performance due to the continuous influence of environmental factors. Their throughputs on Wi-Fi are the most unstable, with the fluctuating range over 50%. While MP-OL achieves the highest total throughput of $7.3Mbps \sim 8Mbps$, its fluctuating range is under 35%. Similar to the results in controlled networks, MP-OL achieves the most significant throughput improvement among all tested MPTCP variants on both Wi-Fi



(a) From 10:00 AM to 11:59 AM



(b) From 10:00 PM to 11:59 PM

Fig. 12. Tests in real network during different periods of the day, files are downloaded from a cloud-based server using both Wi-Fi and 4G.

link and 4G link. Additionally, MP-OL also shows better stability and has smaller throughput fluctuation on both the whole connection and the subflows.

Jointly considering the results in Sections V-D, V-E and V-F, we find that MP-OL can adapt well to various change modes of the network environment, perform well in real network, and is basically excelled than the other MPTCP variants. This proves that MP-OL obtains good generalization capability by reasonably setting the monitoring parameters n and d.

VI. Conclusion

In this paper, we analyzed the congestion control of multipath transmission from the perspective of online learning, and proposed MP-OL, a phased online multipath congestion control algorithm. In our proposal, with a MAB-based model and clever design of the utility function, we enable subflows and flows to achieve significant performance benefits through cooperative game. We also adopted a near-optimal rate control method, which improves the algorithm efficiency and reduces the computational complexity.

We implemented MP-OL in the Linux kernel. Experimental results proved that MP-OL overcomes the limitations of the existing online approaches. We also demonstrated the advantages of our scheme in fairness, link utilization, resilience to non-congestion loss, and adaptability to unstable network conditions over the conventional approaches. Our evaluation of MP-OL's performance in real network motivates further evaluation of MP-OL in some other network scenarios, such as data center and time-sensitive network. Additionally, our current scheme is designed centering around online learning that has to sacrifice a fraction of convergence speed and accuracy to ensure flexibility and lightweight. Thus, the tradeoff between efficiency and effectiveness also deserves further study.

References

[1] M. Li *et al.*, "Multipath transmission for the Internet: A survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 4, pp. 2887–2925, 4th Quart., 2016.

[2] A. Ford, C. Raiciu, M. J. Handley, O. Bonaventure, and C. Paasch, "TCP extensions for multipath operation with multiple addresses," IETF, RFC 6824, 2013. Accessed: Apr. 2022. [Online]. Available: https://www.ietf.org/rfc/rfc6824.txt

[3] C. Raiciu, M. Handley, and D. Wischik, "Coupled congestion control for multipath transport protocols," IETF, RFC 6356, 2013. Accessed: Apr. 2022. [Online]. Available: https://www.ietf.org/rfc/rfc6356.txt

[4] R. Khalili, N. Gast, M. Popovic, and J.-Y. Le Boudec, "Opportunistic linked-increases congestion control algorithm for MPTCP," IETF, Internet-Draft draft-khalili-mptcp-congestion-control-05, 2014. Accessed: Dec. 2021. [Online]. Available: https://www.ietf.org/archive/id/draft-khalili-mptcp-congestion-control-05.txt

[5] R. Khalili, N. Gast, M. Popovic, and J.-Y. Le Boudec, "MPTCP is not Pareto-optimal: Performance issues and a possible solution," *IEEE/ACM Trans. Netw.*, vol. 21, no. 5, pp. 1651–1665, Oct. 2013.

[6] 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.

[7] Z. Xu, J. Tang, C. Yin, Y. Wang, and G. Xue, "Experience-driven congestion control: When multi-path TCP meets deep reinforcement learning," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1325–1336, Jun. 2019.

[8] E. F. G. M. Beig, P. Daneshjoo, S. Rezaei, A. A. Movassagh, R. Karimi, and Y. Qin, "MPTCP throughput enhancement by Q-learning for mobile devices," in *Proc. IEEE 20th Int. Conf. High Perform. Comput. Commun. IEEE 16th Int. Conf. Smart Cit IEEE 4th Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS)*, 2018, pp. 1171–1176.

[9] 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.

[10] 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*, 2019, pp. 75–77.

[11] 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.

[12] M. S. Kim, "Intelligent reinforcement-learning-based network management," IETF, Internet-Draft draft-kim-nmrg-rl-05, 2019. Accessed: Apr. 2022. [Online]. Available: https://www.ietf.org/archive/id/draft-kim-nmrg-rl-05.txt

[13] K. Winstein and H. Balakrishnan, "TCP ex machina: Computer-generated congestion control," in *Proc. ACM Conf. Commun. Archit. Protocols Appl. (SIGCOMM)*, 2013, pp. 123–134.

[14] Z. M. Fadlullah, F. Tang, B. Mao, J. Liu, and N. Kato, "On intelligent traffic control for large-scale heterogeneous networks: A value matrix-based deep learning approach," *IEEE Commun. Lett.*, vol. 22, no. 12, pp. 2479–2482, Dec. 2018.

[15] N. Jay, N. H. Rotman, B. Godfrey, M. Schapira, and A. Tamar, "A deep reinforcement learning perspective on Internet congestion control," in *Proc. 36th Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 3050–3059.

[16] B. He *et al.*, "DeepCC: Multi-agent deep reinforcement learning congestion control for multi-path TCP based on self-attention," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 4, pp. 4770–4788, Dec. 2021.

[17] M. Dong *et al.*, "PCC vivace: Online-learning congestion control," in *Proc. 15th USENIX Symp. Netw. Syst. Des. Implement. (NSDI)*, 2018, pp. 343–356.

[18] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in NLP," in *Proc. 57th Conf. Assoc. Comput. Linguist. (ACL)*, 2019, pp. 3645–3650.

[19] F. Kaup, M. Wichtlhuber, S. Rado, and D. Hausheer, "Can Multipath TCP save energy? A measuring and modeling study of MPTCP energy consumption," in *Proc. 40th IEEE Conf. Local Comput. Netw. (LCN)*, 2015, pp. 442–445.

[20] Z. Xu *et al.*, "Experience-driven networking: A deep reinforcement learning based approach," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2018, pp. 1871–1879.

[21] M. Dong, Q. Li, D. Zarchy, P. B. Godfrey, and M. Schapira, "PCC: Re-architecting congestion control for consistent high performance," in *Proc. 12th USENIX Symp. Netw. Syst. Des. Implement. (NSDI)*, 2015, pp. 395–408.

[22] A. Vaswani *et al.*, "Attention is all you need," in *Proc. 31st Conf. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 5998–6008.

[23] M. E. Peters *et al.*, "Deep contextualized word representations," in *Proc. Conf. North Amer. Ch. Assoc. Comput. Linguist. (NAACL)*, 2018, pp. 2227–2237.

[24] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Ch. Assoc. Comput. Linguist. (NAACL)*, 2019, pp. 4171–4186.

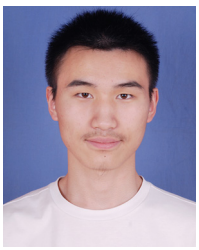
[25] D. R. So, Q. V. Le, and C. Liang, "The evolved transformer," in *Proc. 36th Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 5877–5886.

[26] Dimensional Research, "Artificial intelligence and machine learning projects are obstructed by data issues," Austin, TX, USA, Alegion, White Paper, 2019. Accessed: Apr. 2022. [Online]. Available: https://content.alegion.com/dimensional-researchs-survey
[27] "Multipath TCP in ns-3." Accessed: Apr. 2022. [Online]. Available: https://www.nsnam.org
[28] J. Y. Yu and S. Mannor, "Piecewise-stationary bandit problems with side observations," in *Proc. 26th Int. Conf. Mach. Learn. (ICML)*, 2009, pp. 1177–1184.
[29] H. Luo, A. Agarwal, and J. Langford, "Efficient contextual bandits in non-stationary worlds," in *Proc. 31st Conf. Learn. Theory (COLT)*, 2018, pp. 1739–1776.
[30] Q. Wu, N. Iyer, and H. Wang, "Learning contextual bandits in a non-stationary environment," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval (SIGIR)*, 2018, pp. 495–504.
[31] F. Liu, J. Lee, and N. Shroff, "A change-detection based framework for piecewise-stationary multi-armed bandit problem," in *Proc. 32nd AAAI Conf. Artif. Intell. (AAAI)*, 2018, pp. 3651–3658.
[32] Y. Cao, Z. Wen, B. Kveton, and Y. Xie, "Nearly optimal adaptive procedure with change detection for piecewise-stationary bandit," in *Proc. 22nd Int. Conf. Artif. Intell. Stat. (AISTATS)*, 2019, pp. 418–427.
[33] J. Nash, "Equilibrium points in *n-person* games," *Proc. Nat. Acad. Sci. United States Amer.*, vol. 36, no. 1, pp. 48–49, 1950.
[34] J. Nash, "Two-person cooperative games," *Econometrica*, vol. 21, no. 1, pp. 128–140, 1953.
[35] E. Even-Dar, Y. Mansour, and U. Nadav, "On the convergence of regret minimization dynamics in concave games," in *Proc. 41st Annu. ACM Symp. Theory Comput. (STOC)*, 2009, pp. 523–532.
[36] A. Walid, Q. Peng, J. Hwang, and S. Low, "Balanced linked adaptation congestion control algorithm for MPTCP," IETF, Inernet-Draft draft-walid-mptcp-congestion-control-04, 2016. Accessed: Apr. 2022. [Online]. Available: https://www.ietf.org/archive/id/draft-walid-mptcp-congestion-control-04.txt
[37] Y. Cao, M. Xu, and X. Fu, "Delay-based congestion control for multipath TCP," in *Proc. 20th IEEE Int. Conf. Netw. Protocols (ICNP)*, 2012, pp. 1–10.
[38] "Multipath TCP in the Linux kernel." Accessed: Apr. 2022. [Online]. Available: https://www.multipath-tcp.org
[39] T. Viernickel, A. Frömmgen, A. Rizk, B. Koldehofe, and R. Steinmetz, "Multipath QUIC: A deployable multipath transport protocol," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2018, pp. 1–7.
[40] "Multipath TCP—Linux kernel implementation." Accessed: Apr. 2022. [Online]. Available: https://github.com/esnet/iperf
[41] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance," in *Proc. ACM Conf. Commun. Archit. Protocols Appl. (SIGCOMM)*, 1994, pp. 24–35.
[42] S. Ha, I. Rhee, and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," *ACM SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, 2008.
[43] R. Jain, "The art of computer systems performance analysis: Techniques for experimental design, measurement, simulation and modelling," *SIGMETRICS Perform. Eval. Rev.*, vol. 19, no. 2, pp. 5–11, 1991.
[44] 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./Aug. 2020.
[45] 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.
[46] W. John, S. Tafvelin, and T. Olovsson, "Trends and differences in connection-behavior within classes of Internet backbone traffic," in *Proc. 9th Int. Conf. Passive Act. Netw. Meas. (PAM)*, 2008, pp. 192–201.

Rui Zhuang (Graduate Student Member, IEEE) received the bachelor's degree from the Department of Information Engineering, Hefei University of Technology in July 2019. She is currently pursuing the doctor's degree with the School of Cyber Science and Technology, University of Science and Technology of China. Her research interests include future Internet architecture design, transmission optimization, and data center network.

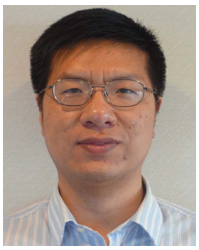
Jiangping Han received the bachelor's and doctor's degrees from the Department of Electronic Engineering and Information Science, University of Science and Technology of China (USTC), in 2016 and 2020, respectively. From November 2019 to October 2021, She was a Visiting Scholar with the School of Computing, Informatics, and Decision Systems Engineering, Arizona State University. He is currently a Postdoctoral 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 and the doctor's degree from the Department of Electronic Engineering and Information Science, University of Science and Technology of China (USTC) in 2003 and 2007, respectively. From May 2012 to May 2013, he was a Postdoctoral 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 serves on the editorial board of several journals, including the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, and the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT. He has also served as a (Lead) Guest Editor for many reputed journals/magazines, including IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, *IEEE Communications Magazine*, and IEEE NETWORK. He is an IET Fellow.



Jian Li (Member, IEEE) received the bachelor's degree from the Department of Electronics and Information Engineering, Anhui University in 2015, and the doctor's degree from the Department of Electronic Engineering and Information Science, University of Science and Technology of China (USTC) in 2020. From November 2019 to November 2020, he was a Visiting Scholar with the Department of Electronic and Computer Engineering, University of Florida. He is currently a Postdoctoral Researcher with the School of Cyber Science and Technology, USTC. His research interests include wireless communications, satellite networks, and next-generation Internet.

David S. L. Wei (Life Senior Member, IEEE) received the doctor's degree in computer and information science from the University of Pennsylvania in 1991. From May 1993 to August 1997 he was on the Faculty of Computer Science and Engineering with the University of Aizu, Japan (as an Associate Professor and then a Professor). He is currently a Professor of Computer and Information Science Department with Fordham University. He has authored and coauthored more than 120 technical papers in various archival journals and conference proceedings. His research interests include cloud computing, big data, Internet of Things, and cognitive radio networks. He was a Guest Editor or a Lead Guest Editor for several special issues in the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, THE IEEE TRANSACTIONS ON CLOUD COMPUTING, and the IEEE TRANSACTIONS ON BIG DATA. He also served as an Associate Editor of IEEE TRANSACTIONS ON CLOUD COMPUTING from 2014 to 2018 and an Associate Editor of *Journal of Circuits, Systems and Computers* from 2013 to 2018.



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

Qibin Sun (Fellow, IEEE) received the doctor's degree from the Department of Electronic Engineering and Information Science, 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 as well as academic research projects in the area of media security and image and video analysis. He was the Head of Delegates of Singapore in ISO/IEC SC29 WG1(JPEG). He worked as a Research Scientist with 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.

Jun Lu received the bachelor's degree from southeast university in 1985 and the master's degree from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), in 1988, where he is currently a Professor with the School of Cyber Science and Technology and the Department of EEIS. His research interests include theoretical research and system development in the field of integrated electronic information systems. He is an Academician of the Chinese Academy of Engineering.