MPLibra: Complementing the Benefits of Classic and Learning-based Multipath Congestion Control

Hebin Yu, Jiaqi Zheng, Zhuoxuan Du, Guihai Chen
State Key Laboratory for Novel Software Technology, Nanjing University, China

***Abstract*—Multipath TCP (MPTCP) is a burgeoning transport protocol which enables the server to split the traffic across multiple network interfaces. Classic MPTCPs have good friendliness and practicality such as relatively low overhead, but are hard to achieve consistent high-throughput and adaptability, especially for the ability of flexibly balancing congestion among different paths. In contrast, learning-based MPTCPs can essentially achieve consistent high-throughput and adaptability, but have poor friendliness and practicality. In this paper, we proposed MPLibra, a combined multipath congestion control framework that can complement the advantages of classic MPTCPs and learning-based MPTCPs. Extensive simulations on NS3 show that MPLibra can achieve good performance and outperform state-of-the-art MPTCPs under different network conditions. MPLibra improves the throughput by 40.5% and reduces the file download time by 47.7% compared with LIA, achieves good friendliness and balances congestion timely.**

I. Introduction

Smart mobile devices and high-performance servers are increasingly equipped with multiple network interfaces to improve throughput or provide additional backup path for reliability. For example, mobile devices usually can use WiFi and 4G simultaneously, while the servers in production data centers usually take parallel paths to perform load balancing. MPTCP [2], as a multipath transport protocol, splits the traffic across multiple network interfaces, which was maintained by IETF working group [10]. With the increasing popularity of MPTCP [3], improving the multipath transmission performance [35] [24] is put into the spotlight and the most important one is designing novel congestion control algorithm.

Compared with single-path TCP, MPTCP has higher requirements for the congestion control algorithms. The objective of MPTCP [29] is significantly different from TCP and MPTCP aims to (1) **improve throughput:** an MPTCP flow should achieve no less throughput than that a TCP flow would achieve on the best of its paths; (2) **do no harm:** an MPTCP flow should not aggressively harm other flows in terms of throughput achieved; (3) **balance congestion:** an MPTCP flow should shift traffic from the congested subpath to the uncongested subpath as soon as possible.

Existing congestion control algorithm for MPTCP can be divided into classic MPTCPs and learning-based MPTCPs, depending on whether the rate adjustment is based on hard-wired mapping decisions or machine intelligence. LIA [32], as a representative of classic MPTCPs, couples the congestion windows of all its subflows when increasing one subflow's rate. The objective aims to achieve the three goals mentioned above. OLIA [19] further improves LIA in terms of pareto optimality and friendliness. BALIA [27] points out that OLIA may hurt adaptability and aims to strike a good balance between LIA and OLIA. State-of-the-art classic MPTCPs always perform fixed AIMD-based rate adjustment. Though good friendliness and practicality offered, they are essentially hard to achieve consistent high-throughput and adaptability, especially taking the path diversity to mitigate the transient congestion. As for the learning-based MPTCPs, due to the great potential to adapt to changing network conditions, they have been applied to congestion control algorithm to improve performance. Specifically, online learning-based MPTCPs such as MPCC [11] adjusts the sending rate based on the real-time feedback from the networks but it suffers from slow convergence and high overhead. The RL(Reinforcement Learning)-based MPTCPs such as SmartCC [11] and DRL-CC [34] train models in simulated networks and hope the well-trained models have learnt the policies that can achieve consistent high-throughput. However, they are still fraught with a series of practical problems such as large variance of subpath delay, slow training speed, inflexible decision model and poor friendliness. Hence, existing learning-based MPTCPs are far from deployment in practice.

In this paper, we propose MPLibra, a multipath congestion control framework that combines the advantages of classic and learning-based MPTCPs together. The RL-agent developed in our framework can decoupledly make decisions for each subflow with the objective of maximizing the subflow-level benefits. At the same time, we use classic coupled MPTCP to guarantee connection-level friendliness and the performance lower bound. By simultaneously borrowing the wisdom of both classic and learning-based MPTCPs, we can periodically obtain two candidate decisions, respectively. Our utility function-based evaluation mechanism is used to select a more proper one on the fly as the final rate.

Our first contribution is that we unveil the performance of state-of-the-art classic and learning-based MPTCPs and summarize the limitations on consistent high-throughput, friendliness, adaptability and practicality. Specifically, We find that learning-based MPTCPs achieves 43.8% higher throughput and reduces the convergence time by 86.7% compared with classic MPTCPs. However, The overhead of the learning-based method is very high while the overhead of the classic methods are almost negligible. Further more, learning-based MPTCPs

978-1-6654-4131-5/21/$31.00 ©2021 IEEE

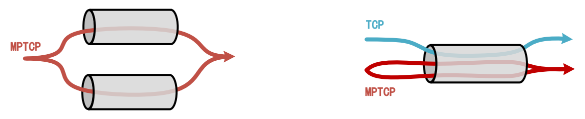
have problems on large variance of subpath delay, slow training speed, inflexible decision model and poor friendliness.

Our second contribution is that, we are the first work to complement the benefits of both learning-based and classic MPTCPs. We present an unified multipath congestion control framework — MPLibra, which mainly includes exploration, evaluation and exploitation stage. During these three stages, MPLibra can periodically select a more proper rate from two candidate decisions derived from classic and learning-based MPTCPs and can achieve consistent high-throughput, adaptability, friendliness and practicality.

Our third contribution is a comprehensive performance evaluation of MPLibra under a variety of scenarios. Extensive NS3 simulations show that, compared with state-of-the-art, MPLibra can (i) improve the throughput by 40.5% and reduce the file download time by 30.0%; (ii) Reduce the queuing delay by 46.7%; (iii) guarantee both friendliness and convergence; (iv) timely balance congestion among different subpaths.

II. Motivation

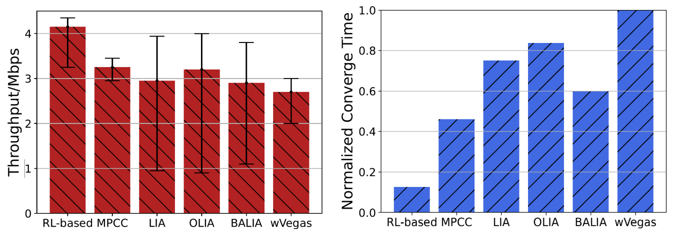
In this section, we enumerate the limitations of the state-of-the-art multipath congestion control methods in terms of consistent high-throughput, friendliness, adaptability and practicality. The consistent high-throughput is a key motivation to design MPTCP as it can essentially provide opportunities to use more than one path. We hope a modern MPTCP should maintain consistent high-throughput under a variety of scenarios such as LTE, Wi-Fi and wired networks. The friendliness requires that the MPTCP flow and the TCP flow can obtain a fair share when they compete the bandwidth on one bottleneck link. As for the adaptability, it indicates the ability of balancing congestion among different paths and to some extent can characterize the convergence speed when the network changes happen. Practicality takes into account the overhead and other practical issues especially when applying learning-based methods to MPTCP.



(a) One MPTCP flow passes through 2-link networks (b) One MPTCP flow and one TCP flow compete on 1-link networks

Fig. 1: Evaluated 1- and 2-link networks.

Classic MPTCPs have good friendliness and practicality, but are hard to achieve consistent high-throughput and adaptability. Classic MPTCPs such as LIA [29], OLIA [19], BALIA [27] and wVegas [5] can theoretically guarantee connection-level friendliness. Besides, they has relatively high practicality due to not involving the training complexities and overhead compared with learning-based MPTCPs. Although the good friendliness and practicality offered, they still cannot achieve consistent high-throughput and good adaptability. We first compared the average throughput of the state-of-the-art MPTCPs and the used topology is shown in Fig.1(a). In this



(a) Throughput (b) Normalized Converge Time

Fig. 2: Performance Comparison

set of experiments, the capacities of two links are both set to be 5 Mbps, the buffer size is an integer randomly chosen from [10, 100] and the loss rate ranges from 0.0% to 0.5%. Each data point is an average of at least 20 runs and the flow duration is 100s. From Fig.2(a), we can observe that classic MPTCPs fail to maintain consistent high-throughput. The reason is that classic loss-based MPTCPs use loss as an indicator of congestion, wrongly halving their congestion windows due to non-congestion induced losses, and classic delay-based MPTCPs such as wVegas fail to fully utilize the link capacity since the delay estimation may be not accurate enough especially in the presence of packet losses [26]. What's more, due to the friendliness concern, classic MPTCPs increase the congestion window very slowly even though a lot of vacant capacity is available and no competing flows exist. Specifically, we perform an experiment with a representative MPTCP — LIA — to show the limitations on time-varying link capacities. Initially, the capacities of two links are 4 Mbps, which are last for 70s. Right then, they increase simultaneously to 8 Mbps and last for 70s again. Finally the capacities of two links decrease to 4 Mbps. According to Fig. 3, we can observe the limited adaptability: from 70s to 100s, the second subflow takes about 30s to fully utilize the link capacity. When the capacity halves at 140s, it also takes 20s to converge. This is because, in the design of LIA, a larger congestion window for one subflow significantly slows down the increase of that for the other subflow, leading to the limited adaptability. We further measure the convergence time for state-of-the-art MPTCPs. Here the convergence time of a MPTCP starts from the rate adjustment and ends up with the steady state of both subflows. Fig. 2(b) shows the normalized average converge time which indicates that classic MPTCPs suffer from slow convergence.

TABLE I: Overhead comparison

Scheme	Avg. CPU Utilization	Avg. Memory Utilization
Classic	1.6 %	0.1 %
RL-based	100 %	2.2 %
OL-based	88.7 %	10.1 %

Learning-based MPTCPs can essentially achieve consistent high-throughput and adaptability, but have poor friendliness and practicality. Current learning-based MPTCPs can be broadly divided into two categories: online learning (OL)-based and reinforcement learning (RL)-based MPTCP. The OL-based method MPCC [11] timely calculates the gradient of

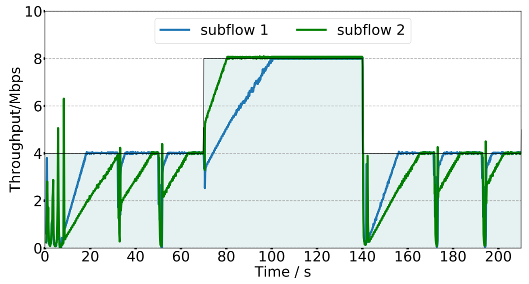
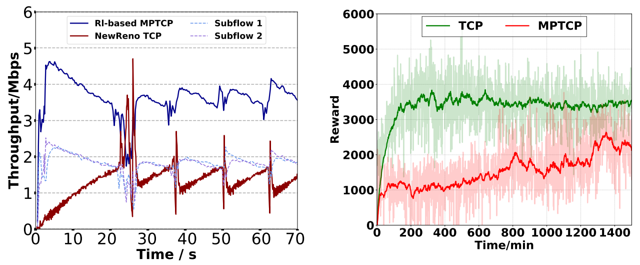


Fig. 3: Illustration of throughput variations for LIA with two subpaths, where the capacities of two links increase from 4 to 8 Mbps at 70s and decrease from 8 to 4 Mbps at 140s.



(a) Unfriendliness (b) Training comparison

Fig. 4: Issues when RL meets MPTCP

the utility function in an online manner and adjusts the sending rate accordingly. However, the online adjustment strategy leads to high overhead as shown in Tab. I and prolonged convergence time as shown in Fig. 2(b) since it doesn't have priori knowledge and has to perform consistent fine-grained trials at the beginning. Note that we measure the overhead of the Linux kernel implementation of NewReno [9], Vivace [7] and Aurora [18] to represent the classic, OL-based and clean slate RL-based MPTCPs, respectively. Unlike OL-based MPTCP, the clean slate RL-based MPTCP trains an offlined RL-agent in advance and then applies this experienced agent to the network environments. Though the clean slate RL-based MPTCPs [22], [34] can fastly converge to the steady state, they suffer from a series of issues below. The most notorious one is poor friendliness. We can observe from Fig. 4(a) that a clean slate RL-based MPTCP with two subflows and a TCP NewReno flow compete together, where the network topology is shown in Fig. 1(b) and the reward function is set to be $\log \sum x_i$ (x_i is the i_{th} subflow's throughput). As for practicality, we compare the efficiency of training a TCP RL agent and an MPTCP RL agent through an experiment. We train an MPTCP agent with two subflows and a TCP agent respectively. Both of them use two hidden layers with 256 neurons and the training algorithms used are DDPG [23]. Through tunning the link parameters such as link capacity, delay and random loss rate, we make the maximum accumulative rewards of these two scenarios the same. As shown in Fig. 4(b), training an agent of 2-subflow MPTCP is much slower than training a single-path TCP agent. The reason is that though the action and state space grow linearly with the increase of the number of subflows, the mapping between the action and the state space grows exponentially which makes it more complex for neuron network to fit. What's more, during the training procedure, the subflows could influence each other and produce noises, leading to increased training complexities. Furthermore, the clean slate RL-based MPTCPs first train the models offline and then deploy them online. This means that, once the number of subflows for MPTCP in the trained model is fixed, they are difficult to change. In general, MPTCP may have any number of subflows and they can be activated and deactivated dynamically due to the changed network conditions. To handle this, LSTM [34] is used, while it makes the training process more complicated and incurs more overhead. Furthermore, the RTT among subpaths may have different orders of magnitude. For example, the RTT of one subpath using WiFi is around 100-200 ms, while the RTT of another subpath using LTE is around 40-60 ms. Existing clean slate RL-based MPTCPs set one unified monitor interval(MI) for all subflows, one agent for one MPTCP. A long MI will lose the flexibility to adapt to the changing network conditions of the short RTT paths and a short MI will be not sufficient enough to present the current network status of the long RTT paths well.

The performance of classic and learning-based MPTCPs are summarized in Tab. II. We find that they have complementary advantages, which motivates us to design a combined multi-path congestion control framework and promisingly overcome the drawbacks above.

TABLE II: Performance comparisons

Goals	Classic	MPCC	RL-based	MPLibra
High-Throughput	Poor	Medium	Good	Good
Friendliness	Good	Good	Poor	Good
Adaptability	Medium	Medium	Good	Good
Practicality	Good	Poor	Poor	Good

III. Overview

MPLibra is an unified multipath congestion control framework that outputs two candidate decisions from both classic and RL-based MPTCPs periodically. Through our utility function-based evaluation mechanism, MPLibra can dynamically pick one proper sending rate from two decisions.

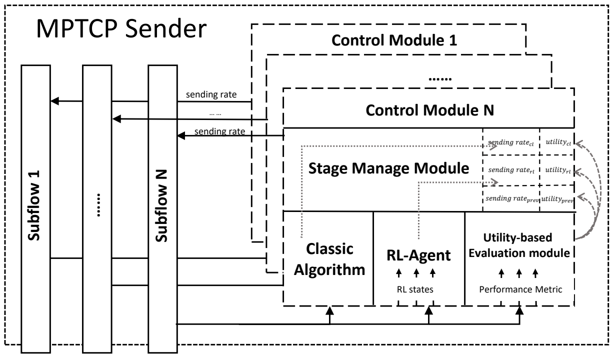


Fig. 5: MPLibra overview

Fig. 5 illustrates the main building blocks of an MPLibra sender. Each subflow corresponds to an individual control module. A control module consists of the RL-agent, classic MPTCP algorithm, utility-based evaluation module and stage manage module. Intuitively, the RL-agent periodically determines a candidate sending rate that maximizes the throughput for one subflow, while the classic MPTCP algorithm also outputs a candidate sending rate aiming to guarantee friendliness. The utility-based evaluation module tries two candidate sending rate one by one for a period, compares the connection-level benefits and finally determines a proper sending rate. Specifically, the utility-based evaluation module calculates the utility value with respect to the real feedbacks such as loss rate and latency gradient from the networks. On one hand, MPLibra can rely on the well-trained RL module to mitigate the misjudgment for the congestion signals, maintain consistent high-throughput and quickly adapt to the network condition changes. On the other hand, when there are competing flows, increasing the rate unilaterally may lead to a rapid increase in delay and loss rate, thus reducing the utility value. In this case, the more prudent final rate given by the classic algorithm considering friendliness is more likely to be chosen. The stage manage module controls the stage transition of one of MPLibra's subflow according to the sending rate generated by RL-agent and classic MPTCP algorithm and the utility value obtained by utility module. Every subflow starts with the exploration stage. In the exploration stage, the RL method and the classic method respectively generate actions based on the current network state. The evaluation stage is used to evaluate the transient performance of the two actions generated in last the phase and select a better one of them. Based on the utility value of each action derived in the evaluation phase, the exploitation stage can exploit the action with the higher utility value.

The detailed mechanism of MPLibra's stage transition can be shown in Fig. 6. According to the RTT of the subflow, MPLibra divides time into subflow moniter intervals (sMI).

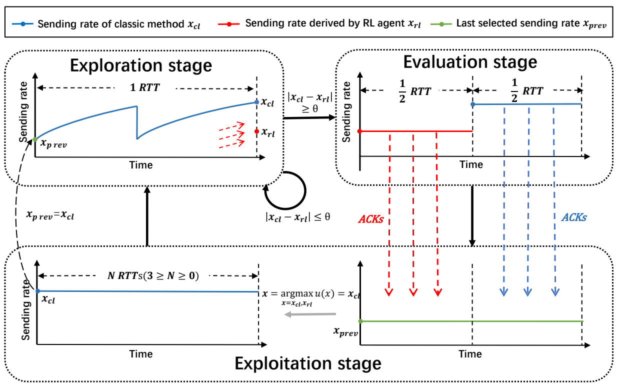

Sending rate of classic method x_{cl}
Sending rate derived by RL agent x_{rl}
Last selected sending rate x_{prev}
Exploration stage
Sending rate
1 RTT
x_{cl}
x_{rl}
$x_{p\,rev}$
Time
$|x_{cl} - x_{rl}| \geq \theta$
Evaluation stage
Sending rate
$\frac{1}{2}$ RTT
$\frac{1}{2}$ RTT
Time
$x_{p\,rev}=x_{cl}$
$|x_{cl} - x_{rl}| \leq \theta$
ACKs
ACKs
Sending rate
N RTTs$(3 \geq N \geq 0)$
x_{cl}
$x = \operatorname{argmax}_{x=x_{cl},x_{rl}} u(x) = x_{cl}$
x_{prev}
Time
Time
Exploitation stage

Fig. 6: Time-diagram of the Stage Manage Module in Fig.5

Exploration stage At the beginning of this stage, the sending rate for each subflow is set to the base sending rate x_{prev} which is the finally determined rate in the last cycle. The exploration lasts for an RTT and the classic MPTCP adjusts the sending rate in a per-ACK manner during this period. At the end of this period, we can obtain an x_{cl} derived by the classic MPTCP, and at the same time, RL-agent generates its decision x_{rl} according to the statistics it gathered during this stage. Then we compare the divergence of the x_{cl} and x_{rl}, we enter into the evaluation stage if $|x_{cl} - x_{rl}|$ is bigger than a threshold θ which is set to 0.2 $\times x_{prev}$ as default. Otherwise, we set x_{cl} as the base sending rate and re-enter into the exploration stage.
Evaluation stage Once we are in the evaluation stage, it means that the decision made by the RL agent and the classic MPTCP are far apart. So we need to verify two candidate decisions and select a better one. The evaluation stage is divided into two evaluation intervals (EI) and each EI runs at a constant sending rate. An EI is set to half of the RTT to better adapt to highly changing networks and mitigating performance degradation caused by wrong evaluated decisions. Once two candidate rates are evaluated, how can we guarantee that two actions do not interfere with each other since the first evaluated decision might cause the queue buildups and lead to misjudge of the second evaluated action? To mitigate the side effect of the mutual influence, we set the sending rate to the smaller one between x_{cl} and x_{rl} in the first EI, and in the second EI, the sending rate is set to the larger one.
Exploitation stage In this stage, we firstly need to collects the feedbacks corresponding to the enforcing sending rates in the last evaluation stage and this lasts for two EIs. Meanwhile, we exploit the sending rate x_{prev} determined in the last cycle. The MPLibra calculates the utility value $u(x_{cl})$ and $u(x_{rl})$ from x_{cl}, x_{rl} and the collected statistics (throughput, RTT, etc.). Then we select the sending rate with a higher utility value as a new base sending rate. We can exploit the new sending rate for N RTTs, where N is a tradeoff between the flexibility and stability and it can be an arbitrary integer between 0 and 3.

IV. MPLibra Design

A. The RL-based module

State: The state is used to reflect the varying network conditions. To well capture the changed network conditions, we picked out several key network features according to prior work [34] [18] [8] [1]: *current sending rate*, *current RTT and the minimum RTT*, *average loss rate of packets* and *average delivery rate*. Furthermore, to characterize the network dynamics, we take the 10 most recent statistics in the history to combine the final state.
Action: Once a new observation is available, the RL-agent derives action to adjust the sending rate. For more fine-grained control of the sending rate, we set the adjustments to be the same as that of Orca [1]:

$$cwnd = 2^{\alpha} \times cwnd_{prev}, (-5 \leq \alpha \leq 5)$$

The corresponding sending rate should be:

$$sending\ rate = cwnd/sRTT$$

Training algorithm: The continuous setting of action space doom the inapplicability of value-based reinforcement learning methods such as DQN [25]. We select deep deterministic policy gradient (DDPG) [23] to train our agent for each subflow since it is an advanced actor-critic method [30].

Reward function: The design of reward function determines the training objective and the direction of adjusting the strategy. Hence, it should well present the expectation of our RL-module: trying to maximize throughput and minimize latency/loss. Accordingly, we set the reward function as

$$R = w1 \cdot x/x_{max} - w2 \cdot d/d_{min} - w3 \cdot l$$

where x is the throughput, d is the delay and l is the loss rate for each subflow. The parameters $w1, w2, w3$ are set to be 1, 1 and 10, respectively.

B. Underlying Classic MPTCP Module

The classic MPTCP algorithm can be either of the currently proposed LIA [32], OLIA [19], BALIA [27] or other classic MPTCP algorithms. We choose OLIA in our experiments due to its advantages on friendliness and provable pareto optimality. Each subflow enables OLIA's control when it is in the exploration stage and disable it when it is not. We claim that the underlying classic algorithm module can be replaced whenever there is a better choice. It's worth noting that the current scheme is designed for loss-based classic MPTCP algorithms. We leave it to future work when the more advanced classic MPTCP congestion control algorithms are proposed.

C. Compound algorithm

Algorithm: As described in algorithm 1, there is a daemon that is always listening for subflows to join. If a new subflow is activated, then it will initialize an RL module for this subflow and starts a control loop for it. The control loop is described in algorithm 2. For each control cycle, the subflow does as follows. In the exploration stage, the sending rate of the subflow is set to x_{prev} (line 5). Then, the sending rate x^s_{rl} and x^s_{cl} are derived by the RL-agent and the classic MPTCPs (lines 6-7). If the difference between x^s_{rl} and x^s_{cl} is greater than the threshold th_1, the subflow proceeds to the evaluation stage (lines 8-9).In the evaluation stage, subflow first calculates utility value of the x^s_{prev} and then tries the sending rates by sending data at x^s_{rl} and x^s_{cl} , respectively (lines 11-13). In the exploitation stage, it sets the sending rate to x^s_{prev}, calculates two utility values $u(x_{rl})$ and $u(x_{cl})$ and set the x^s_{prev} to the rate leading to the highest utility in the first estimated RTTs (lines 16-20). Finally, subflow exploits the newly chosen sending rate for N estimated RTTs (lines 21-22). Then it goes back to the exploration stage.

Utility-based Evaluation Module: The utility function essentially determines the adjustment direction of the sending rate. It aims at improving the throughput of the current connection and guarantee fairness between connections at the connection-level at the same time. Unlike the reward function used for the training discussed above, the utility function serves as an online decision to choose a better sending rate from two candidate rates like MPCC [11]. Note that the friendliness a special case of fairness. In this paper, we focus on friendliness — a special fairness between MPTCP flows and TCP Reno flows. Here, we will show how to achieve connection-level utility-maximization and fairness simultaneously through the design of subflow-level utility function. Consider a subflow i of the MPTCP connection a, we take the sending rate of connection a's other subflows $s \neq i$ as constant t^a_s. The utility of the i_{th} subflow of connection a should be:

$$U^a_i = \log(\sum_{s \neq i} t^a_s + x^a_i) - (\sum_{s \neq i} t^a_s + x^a_i) \cdot (\beta \cdot L^a_i + \gamma \cdot \frac{d(RTT_i)}{dT}) \quad (1)$$

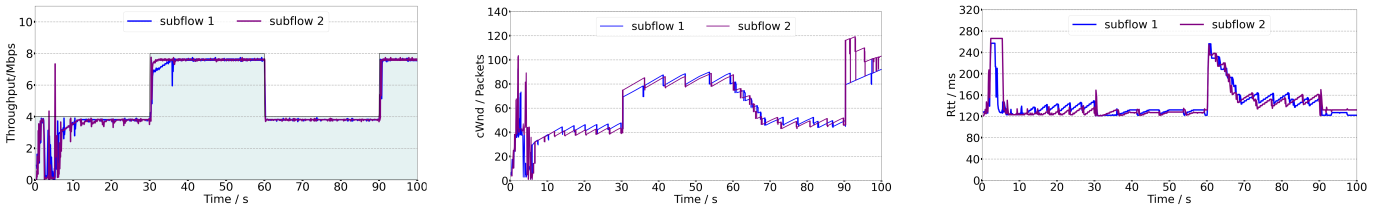
where the L^a_i and $\frac{d(RTT_i)}{dT}$ is the loss rate and latency gradient of the i_{th} subflow for the connection a. Here we set β and γ to be a relatively large value ($\beta = 10$ and $\gamma = 5$) so that the conservative sending rates will be more inclined to be chosen when the sending rate reaches the capacity or the flow is in a competing scenario. Next, we introduce lexicographic max-min fairness [28], and then prove the friendliness and convergence of MPLibra.

Algorithm 1: MPLibra

```
Input: the classic MPTCP $CC_{classic}$; the length of exploitation
       stage $N$; the difference threshold of the decisions $th_1$;
1 while a new subflow $s$ is activated do
2     Initalize RL module $CC_{rl}$;
3     startControlLoop($s, CC_{classic}, CC_{rl}, N, th_1$);
```

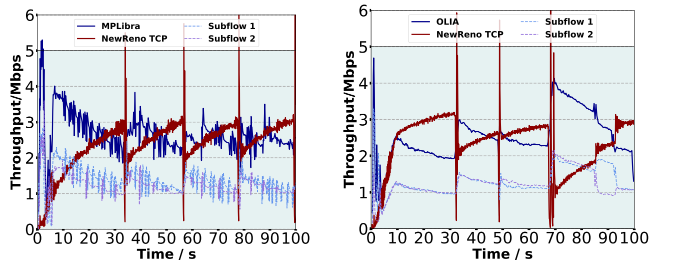
Algorithm 2: The control loop of each subflow

```
Input: subflow $s$; the classic MPTCP $CC_{classic}$; the RL module
       $CC_{rl}$; the length of exploitation stage $N$; the difference
       threshold of the decisions $th_1$;
1  for control cycle $t$ of subflow $s$ do
2      if subflow $s$ is deactivated then
3          break
4      //Exploration stage
5      Initially set the sending rate $x^s_t$ as $x_{prev}$
6      $x^s_{rl} = RL\_Agent(CC_{rl}, S_t)$
7      $x^s_{cl} = Classic\_MPTCP(CC_{classic})$
8      if $|x^s_{cl} - x^s_{rl}| \geq th_1$ then
9          Break, turn into the evaluation stage
10     Reenter Exploration stage
11     //Evaluation stage
12     Try a smaller rate between $x^s_{rl}$ and $x^s_{cl}$ first for one EI.
13     Try the remaining one then for another EI.
14     Collect the performance statistics, calculate utility value
         $u(x^s_{prev})$
15     Turn into the exploitation stage
16     //Exploitation stage
17     if In the first estimated RTT then
18         Send traffic with rate $x^s_{prev}$.
19         Collect the performance statistics corresponding to $x_{cl}$ and
             $x_{rl}$ , respectively.
20         Calculate the utility value $u(x^s_{cl})$, $u(x^s_{rl})$.
21         $x^s_{prev} =$
             $\arg\max_{x^s_{prev}, x^s_{rl}, x^s_{cl}} \{u(x^s_{prev}), u(x^s_{rl}), u(x^s_{cl})\}$
22     for the time in the next N estimated RTTs do
23         Send traffic with rate $x^s_{prev}$.
24     $t = t + 1$
```



(a) Throughput variations (b) Congestion window size variations (c) RTT variations

Fig. 7: MPLibra's behavior under step scenario



(a) MPLibra with two subflows compete with one TCP Reno flow (b) OLIA with two subflow compete with one TCP Reno flow

Fig. 8: MPLibra and OLIA's throughput variations under competing scenarios

Lexicographic Max-Min Fairness (LMMF): Consider allocating the bandwidth of several links to several flows, if an allocation satisfying LMMF, then the bandwidth allocated to the worst off connection, the second-to-worst connection, and so on should all be maximized [11] [28].

Theorem 1. LMMF and Convergence: *For any given networks with k parallel links [11], n MPLibra flows compete with m classic MPTCP flows and MPLibra can achieve an unique equilibrium satisfying LMMF, where the classic MPTCPs can be LIA, OLIA or BALIA.*

The proof can be found in Appendix A. We have proved the intra-protocol fairness between MPLibra flows and the inter-protocol fairness between MPLibra flows and OLIA flows. When the number of the subflows of OLIA or MPLibra flow is 1, they automatically become a NewReno flow and a single path version MPLibra, respectively. Therefore, this actually covers the proof of MPLibra's TCP-friendliness.

V. Evaluation

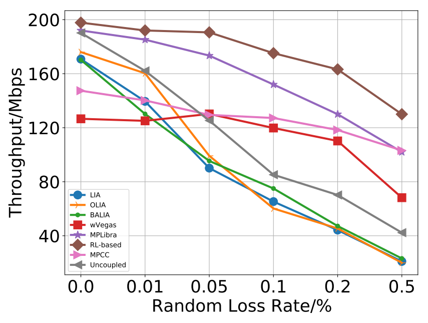
We evaluate MPLibra with extensive simulations compared with state-of-the-art MPTCPs.

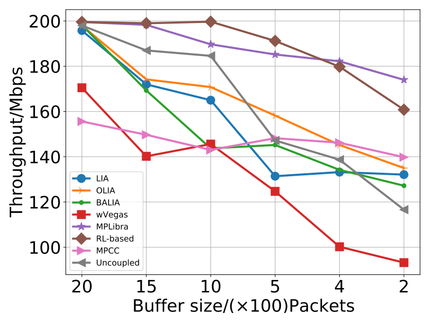
Simulation Setup: Our experiments are based on Kheirkhah's published version [20] of NS3. Specifically, we develop the RL-module of MPLibra in NS3 and implement a key interface `RLInteractModule()` which is responsible for the interaction between the RL-module and socket module. At the same time, `RLInteractModule()` periodically enforces the final sending rate to the socket module. The RL-module is trained on a variety of scenarios to make it more general. In addition, we set the numbers of background TCP flows in the bottleneck link to be a random number during the training procedure.

Benchmark MPTCPs: We compare MPLibra with state-of-the-art classic and learning-based MPTCPs. We implement coupled classic MPTCPs such as LIA [32], OLIA [19], BALIA [27] and wVegas [5] according to standard RFC documents [29] [19] [31] [33]. We also implement an uncoupled classic method by using single path reno for each subflow. As for learning-based MPTCPs, we use online learning-based MPCC [11] and the clean slate RL-based method as two representative benchmarks. For MPCC, We modified some of the hyperparameters to make MPCC in the NS3 for better performance. For clean-slate RL-based method, We implemented a simplified one whose agent can jointly control all subflows that belongs to one MPTCP connection.

Throughput variations under step scenarios. We first emulate a network shown in Fig. 1(a), where the link bandwidth changes dynamically as shown in Fig. 7(a). The blue shaded area indicates the link capacity. Fig. 7 details the throughput, cWnd(congestion window) and RTT variations for MPLibra. At the 30s, the capacity changes to 8 Mbps and accordingly MPLibra increases its sending rate rapidly to grab the available bandwidth. The reason is that the RL-module of MPLibra can see the delay reduction and encourage the rate-increase behavior. At the 60s, the capacity is suddenly halved and this leads to the transient queue accumulation. We observe that the cWnd decreases from about 80 packets to 50 packets shown in Fig. 7(b), which means that the RL-agent observes the delay increase and halves its cWnd. However, classic MPTCP module blindly increase its sending rate until the packet loss happens and thus leads to high latency. Finally, the rate decision from RL-agent is selected with a higher reward through our utility-based evaluation mechanism. This demonstrates that MPLibra can achieve advantages complementary since a well-trained RL-agent module can always correct the behavior of the classic MPTCP module.

Throughput variations under competing scenarios. The network topology under competing scenarios is shown in Fig. 1(b), where the link capacity is 5 Mbps. From Fig. 8, we can observe that the average throughput of MPLibra flow and TCP Reno flow is 2.56 Mbps and 2.23 Mbps, while that of OLIA flow and TCP Reno flow is 2.35 Mbps and 2.22 Mbps, respectively. MPLibra improves the throughput of OLIA by 8.9% and simultaneously does not hurt the throughput of TCP Reno flow. With the help of the classic MPTCP module, MPLibra flow and the TCP Reno flow can converge to a fair steady state in around 20 seconds. Compare to the smoother curves of the OLIA in Fig. 8(b), the throughput variations of


(a) Throughput under different random loss rates


(b) Throughput under shallow buffer

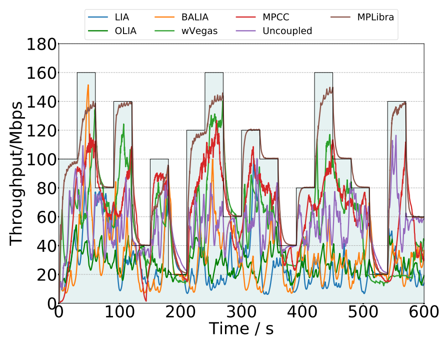
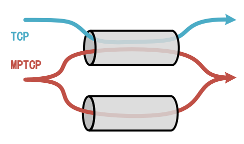
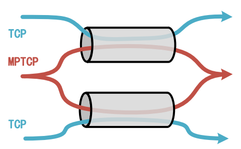

(c) Throughput under changing network conditions

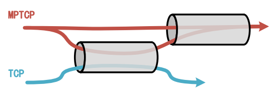
Fig. 9: Throughput comparisons



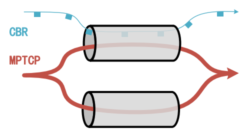
(a) One subflow of MPTCP competes with one TCP Reno flow



(b) Two subflows of MPTCP compete with two different TCP Reno flows



(c) One subflow of MPTCP competes with TCP Reno flow and the other



(d) One subflow of the MPTCP competes with one CBR flow

Fig. 10: Evaluated 2-link networks.

MPLibra experience periodical fluctuation because it needs to select the best one from two candidate decisions. In a word, this demonstrates the importance of the classis MPTCP module in terms of friendliness since the pure learning-based MPTCP always aggresively grab bandwidth and starve the TCP Reno flow as shown in the motivation of Fig. 4(a).

Throughput variations under different random loss rates. We set the different random loss rates for the links in Fig. 1(a) and compare the achieved throughput for benchmark schemes. The results are shown in Fig. 9(a). According to the statistics, the throughput of the clean slate RL-based MPTCP is always high since it has a comprehensive evaluation for the congestion signals and aggressively increases its rate. However, LIA, OLIA and BALIA show their vulnerabilities on consistent high throughput: the throughput is reduced by around 90% when the random loss rate reaches 0.5%. MPCC and delay-based wVegas always under-utilize the link capacity even when the random loss rate is zero. MPLibra presents a good resilience to non-congestion loss and can provide a reasonable tradeoff between high throughput and friendliness.

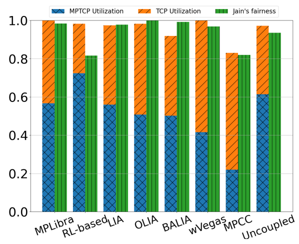
Throughput variations under different buffer sizes. We explore the impact of different buffer sizes on throughput, where the buffer size varies from 200 packets (1/5BDP) to 2000 packets (2BDP). As shown in Fig. 9(b), MPLibra and the clean slate RL-based MPTCP can maintain high throughput even when the buffer size is set to be 200 packets, while the classic MPTCPs can achieve hight throughput only in the presence of deep buffer. This presents the complementary advantages of MPLibra under different buffer sizes.

Throughput variations under changing network characteristics. To present MPLibra's advantages on adaptability, we simulated a network environment based on the topology in Fig.1(a), where the link capacity varies from 10 to 80 Mbps, the link delay varies from 20 to 120ms, the random loss rate varies from 0.01% to 0.05%, and they change every 30s. The blue shaded area in Fig.9(c) indicates the link capacity variations with time. According to the Fig.9(c), we can observe that the throughput of MPLibra is near optimal.

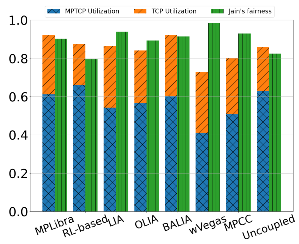
Next we focus on friendliness for four network scenarios. We show the utilization and the Jain's fairness index for each MPTCP connection in Fig.11. Unless specifically stated, all link capacities, delays, random loss rate and buffer sizes are set to 5Mbps, 60ms, 0.001% and 50 packets (1BDP), respectively.

Friendliness: one MPTCP with two subflows competes with one TCP Reno flow on a bottleneck link. The network topology we used is shown in Fig.1(b). The Fig.11(a) shows that uncoupled and the clean slate RL-based MPTCP can achieve around twice as much throughput as TCP Reno flow since their design principles cannot take friendliness into account. Though MPCC does no harm to TCP Reno flows , it fails to maintain the desired throughput. This is because the increased delay and loss rate introduced by TCP Reno flow can also make MPCC proactively reduce the sending rate. wVegas also shows its disadvantages: when competing with loss-based MPTCPs, it achieves less bandwidth than that of TCP Reno flow. Compared with classic loss-based MPTCP, MPLibra can achieve comparable friendliness and link utilization.

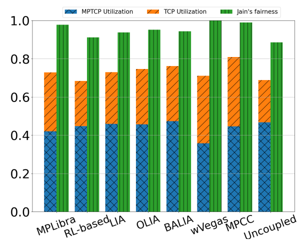
Friendliness: one subflow of MPTCP competes with one TCP Reno flow on a bottleneck link. The network topology used is shown in Fig.10(a). Since the top link has the same capacity as that of the bottom link, ideally MPTCP should shift all traffic to the bottom link. In this scenario, wVegas achieves relatively high fairness index but only about 70% utilization. Nearly all benchmark schemes deviate from the



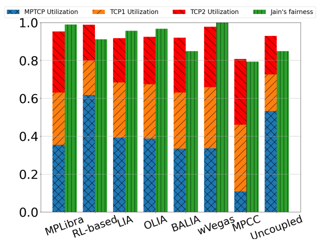
(a) The network topology used is shown Fig.1(b)



(b) The network topology used is shown Fig.10(a)



(c) The network topology used is shown Fig.10(c)



(d) The network topology used is shown Fig.10(b)

Fig. 11: Ratio between the total throughput and capacity and Jain's fairness index [17] comparisons.

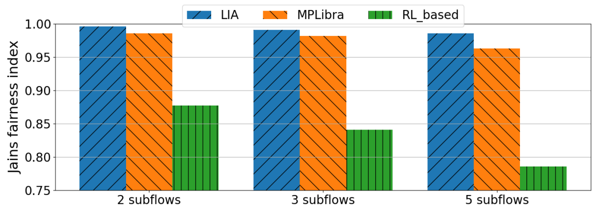
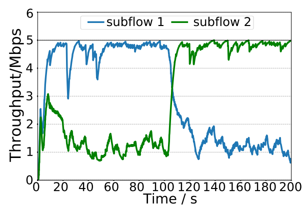
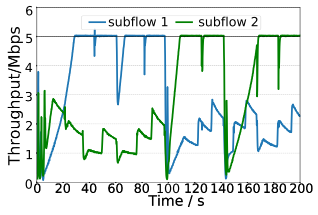


Fig. 12: Friendliness of the MPTCP with arbitrary number of subflows.



(a) MPLibra



(b) BALIA

Fig. 13: Throughput comparisons

ideal equilibrium point and only MPLibra can achieve a good tradeoff between utilization and friendliness.

Friendliness: one subflow of MPTCP competes with TCP Reno flow on the first bottleneck link and this subflow competes with the other subflow on the second bottleneck link. The network topology used is shown in Fig.10(c), where two subflows of MPTCP share a 5 Mbps bottleneck link. In this scenario, if MPTCP cannot well shift the traffic from the bottom link to the top link, it will greatly reduce the overall throughput. From Fig.11(c), we can observe that uncoupled and the clean slate RL-based MPTCPs achieve relatively low link utilization, though their objectives aim to maximize the throughput. At the same time, MPLibraand the coupled MPTCPs perform better.

Friendliness: one subflow of MPTCP competes with the first TCP Reno flow and the other subflow of this MPTCP competes with the second TCP Reno flow. The network topology used is shown in Fig.10(b). Fig.11(d) shows that wVegas is close to the optimal allocation and MPLibra beats benchmark schemes in terms of both overall link utilization and fairness index.

Friendliness: One MPTCP with arbitrary number of subflows competes with TCP Reno flow on a bottleneck link. We test whether MPTCPs' friendliness still holds when they have more than 2 subflows through this experiment. The evaluated topology is similar to the one in Fig.1(b) and the only difference is that the MPTCPs in this experiment have more than 2 subflows. We compare MPLibra's friendliness when it has 2, 3 and 5 subflows with LIA and clean-slate RL-based method. The Fig.12 shows the Jain's fairness indexs of the MPTCP flows and the single path TCP flows and it indicates that MPLibra and LIA can still ensure friendliness but clean-slate RL-based method deviates further from the friendly distribution with the increase of the number of subflows.

Based on the discussions above, we can derive that MPLibra can maintain good performance in terms of link utilization and friendliness under a variety of scenarios. In the following, we evaluate MPLibra's ability of balancing congestion.

Performance on balancing congestion: there is a TCP flow that suddenly joins and terminates. Based on the scenario in Fig.10(b). We enable the regular TCP flow of the bottom link(which competes with MPTCP's subflow2) from 0s to 100s and the one of the top link (which competes with MPTCP's subflow1) from 100s to 200s. The optimal result is that the MPTCP can immediately transfer the traffic from subflow1 to subflow2 at 100s. We plot the MPLibra and BALIA's throughput during the whole process in Fig.13. We observe that both MPLibra and BALIA can well handle this situation since they can complete shifting within 15s.

Performance on balancing congestion: there is a bursty CBR flow. Next we compare the performance of MPLibra and baselines under the scenario from Fig.10(d) that requires fast load balancing ability. The bursty CBR flow sends traffic to the top link at 5Mbps for a random duration of 1s, then it stays off for a random duration of 10s. The most ideal behavior of the MPTCP is sending no traffic to the top link when there is CBR flow and take up the link capacity immediately when the CBR flow is off. Fig.14 shows that the classic methods fail to fastly grab the free capacity when CBR flow is quiet thus leading to their under utilization of the top link. the clean slate RL-based method can flexibly tune its sending rate to adapt to the changing network and the MPLibra can achieve comparable performance. wVegas and MPCC fail to fully utilize both links in this experiment.

Performance on practicality: overhead Since the learning-based methods usually has the problem of high computation

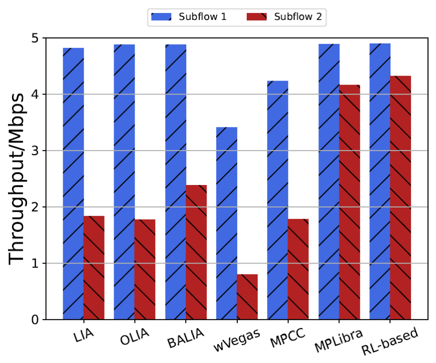


Fig. 14: Performance on balancing the traffic among different subpaths

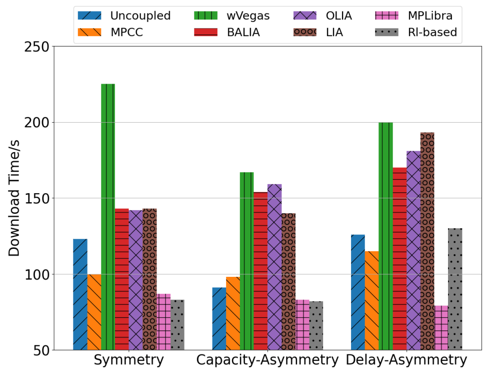


Fig. 15: The download time comparisons of a 75MB file through symmetry and asymmetry networks.

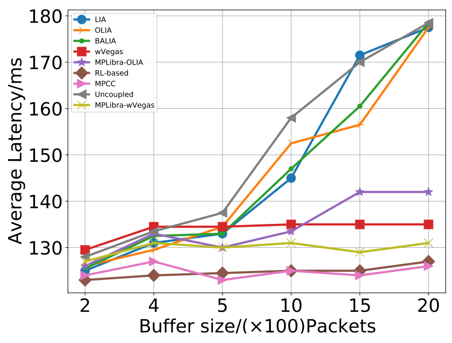


Fig. 16: Average latency under different buffer sizes

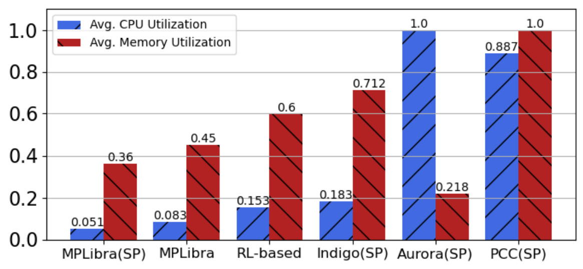


Fig. 17: The normalized overhead comparisons of learning-based methods. The schemes that are marked with ”SP” are single-path methods.

overhead, we focus on comparing the CPU utilization and memory utilization of the energy-extensive learning module of the existing methods in Fig.17. As show in Fig.17, the single path MPLibra achieves much lower CPU and memory utilization compared with other single path learning-based schemes such as PCC, Aurora and Indigo. When the number of subflows expands to 2, MPLibra can still achieve acceptable computation overhead.

There are several reasons to believe that MPLibra can significantly reduce the computation overhead compared with the clean slate RL-based schemes. First, our RL-agent doesn't need to perform on every MI, it only derives action at the end of the exploration stage. Second, because the training complexities are reduced, the agent's network are more potential to be simplified and it will lead to lower cost of the computation.

Performance on practicality: the issues of the existing learning-based methods The design of MPLibra can completely avoid the practical problems of learning-based method mentioned in the Sec. II. Since we arrange RL agents on each subflow, we no longer have to set an unified MI for all subflows and every subflow's MI can be set according to its RTT. Also, we don't suffer the problems of training and the flexible subflow numbers since whenever a new subflow comes in or an old subflow ends, we can safely arrange RL agents that can assure sublfow-level performance according to the number of active subflows.

Performance on file download time. File transfer download time is an practical application-level metric that can better show the transfer efficiency. We simulate the transfer of a 75MB file using different benchmarks. We conduct extensive experiments on symmetry, capacity-asymmetry and delay-asymmetry paths respectively. For symmetry paths, both paths have 5 Mbps capacity, 120 ms latency, 50 packets buffer (1BDP) and 0.05% random loss rate. Fig. 15 shows that the clean slate RL-based MPTCP completes the transfer first and MPLibra follows. For capacity-asymmetry paths, the capacity of path 1 and path 2 are 4 Mbps and 8 Mbps respectively, the other settings are the same as that of the symmetry one. According to Fig. 15, MPLibra also shows high efficiency on file transfer. For delay-symmetry paths, we set the delay of path 1 and path 2 to 60 ms and 120 ms respectively. Note that here the clean slate RL-based MPTCP no longer outperforms benchmark schemes since the length of its monitor interval is fixed and cannot well match the delay asymmetry scenarios. Overall, MPLibra reduces file transfer time by an average of 47.7% compared to LIA.

Performance on low latency. We compare the latency achieved by benchmark schemes on the network topology shown in Fig. 1(a), where the buffer size varies form 10 to 100 packets. The base RTT and the capacities of both links are 120 ms and 5 Mbps, respectively. Fig. 16 shows that the average delay of the classic loss-based MPTCPs becomes larger when the buffer sizes increase. The clean slate RL-based MPTCP and MPCC can minimize the latency since both of their reward functions include the latency. Though the default classic MPTCP module of MPLibra is the loss-based OLIA, MPLibra can significantly reduce the high latency associated with classic loss-based MPTCPs. In addition, to show the flexibility of MPLibra's classic MPTCP module, we also evaluate the MPLibra-wVegas whose classic MPTCP module is wVegas, and Fig. 16 shows that the latency of MPLibra-wVegas without the base latency (120 ms) is at most about half as much as that of MPLibra.

VI. Related Work

Classic MPTCPs: Initially, MPTCPs use an uncoupled mechanism to perform multipath congestion control [16], simply applying TCP NewReno to each subflows. Due to fairness concern, EWTCP [15] adjusts the additive parameters and tries to guarantee connection-level fairness. Later, LIA [32] is proposed to maintain consistent high throughput and fairness. Khalili et al. observe that bandwidth allocation in LIA is not Pareto-optimal due to improper adjustment of congestion window size, and their proposed OLIA [19] can achieve the optimal allocation and friendliness. Furthermore, BALIA [27] attempts to strike a balance between LIA and OLIA. wVegas — a delay-based MPTCP —- extends Vegas [4] to a multipath networks and uses the queuing delay as the congestion signal to perform fine-grained rate decisions. Furthermore, TCP Cubic [12] and BBR [6] are also extended to MPTCP [21] [13] respectively.

Learning-based MPTCPs: In the last few years, several learning-based MPTCPs emerge. Xu et al. [34] use the deep reinforcement learning to solve the multipath congestion control problem for the first time, which integrates LSTM [14] and actor-critic networks for end-to-end model training to reduce the impact of sudden entry and termination of flow in MPTCP environments. Li et al. [17] propose SmartCC to deal with the diversity of multiple paths in a heterogeneous network such as buffer bloat and unideal performance of bandwidth utilization. As the variant of the famous PCC Vivace [7], MPCC [11] propose an online-learning method which use gradient ascent to reach the global optimal point.

VII. Conclusion

In this paper, we design and evaluate a multipath congestion control framework MPLibra which can leverage the advantages of the classic and RL-based MPTCPs. Extensive experiments show that MPLibra outperforms state-of-the-art MPTCPs in a variety of performance metrics such as consistent high-throughput, friendliness to TCP Reno flow and the ability to balance congestion.

Acknowledgement

We would like to thank our shepherd Hulya Seferoglu and anonymous reviewers for their valuable comments on drafts of this paper. This work was supported in part by the China NSF grants (62172206, 61972254, 61802172), China NSF of Jiangsu Province (BK20201248) and Open Fund of PDL (WDZC20205500109).

Appendix A
Proof for Theorem 1

We prove MPLibra's properties on LMMF and convergence by the following three lemmas. Lemma 1 and Lemma 2 indicate that the bandwidth allocation in the equilibrium point satisfies LMMF. Lemma 3 shows that MPLibra can converge to an equilibrium point.

Lemma 1. *MPLibra's utility function tends to select a decision that satisfies LMMF in an equilibrium point.*

Proof. For MPLibra flows, Let x_r stands for the sending rate of a subflow r and let t_s stands for the fixed sending rate of a subflow$s$. We have the second derivative of the utility function of the subflow *r*'s sending rate x^r:

$$\frac{\partial^2(\log(\sum_{s\neq r} t^s + x^r) - \beta\cdot(\sum_{s\neq r} t^s + x^r)\cdot L_i^a)}{\partial(x^r)^2} = -\frac{1}{(\sum_{s\neq r} t_s + x^r)^2} < 0 \tag{2}$$

Therefore, the utility function is strictly concave. According to [11], any equilibrium resulted by such a utility function is LMMF.

□

Lemma 2. *The equilibrium point resulting from LIA, OLIA and BALIA satisfies LMMF.*

The proof of Lemma 2 can be found in [32] [19] [27].

Lemma 3. *MPLibra can converge to an equilibrium point.*

Proof. Here we present the proof that the LMMF equilibrium point can be achieved. Without loss of generality, the following proof assumes that there are no two subflows from one connection share a link since in that case the utility value of each subflow would be the same and the rate adjustment will also be very similar.

When the equilibrium point has not been reached, the rate allocation will move to the equilibrium point. We assume the connection *i* and connection *j* whose subflows are sending traffic on the link *l*. Define C^l as the capacity of link *l* and $x^l = x^i + x^j$ as the traffic of link *l*. Let's discuss in cases:

(i). $x^l < C^l$**:** Under these conditions, there will be no loss rate increase thus classic MPTCPs will increase its sending rate in AI manner and we expect the RL-based method can utilize the capacity fastly. Finally, MPLibra will prefer a faster one of classic MPTCPs and RL-based to choose. Then it will lead to the increase of x_l and move to equilibrium.

(ii). $x^l > C^l$**:** When the traffic exceeds capacity, classic MPTCPs will halve its *cwnd* to half and RL-based will also tends to reduce sending rate since it sees the leap of its loss rate. Then it will leads to the decrease of x^l and move to equilibrium.

(iii). $x^l = C^l$ **but** $t^i > t^j$**:** As the equation 2 suggests, the derivative of the utility function of connection *i*'s subflow is strictly higher. As a result, *i*'s subflow will prefer more aggressive decisions compare to *j*'s subflow and the difference $|x^i - x^j|$ will be increasingly smaller and finally move to a allocation that satisfies LMMF.

When the equilibrium point has been reached, the rate allocation will stabilize within a certain range. With the guidance of the utility function, the total traffic x^l of link l will be stable in the range $(C^l, C^l(1+\frac{1}{\beta-2})]$ [11] from some point in time onwards. The classic MPTCPs' stability has also been prooved in [32] [19] [27]. Although the RL module of MPLibra is not stable in some untrained scenarios, it can be compensated by the theoretical advantages of the classic MPTCP in stability. □

References

[1] S. Abbasloo, C.-Y. Yen, and H. J. Chao. Classic meets modern: A pragmatic learning-based congestion control for the internet. In *ACM SIGCOMM*, pages 632–647, 2020.

[2] S. Barré, C. Paasch, and O. Bonaventure. Multipath tcp: from theory to practice. In *International conference on research in networking*, pages 444–457. Springer, 2011.

[3] O. Bonaventure, C. Paasch, G. Detal, et al. Use cases and operational experience with multipath tcp. *RFC 8041*, 2017.

[4] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson. Tcp vegas: New techniques for congestion detection and avoidance. *SIGCOMM Comput. Commun. Rev.*, 24(4):24–35, Oct. 1994.

[5] Y. Cao, M. Xu, and X. Fu. Delay-based congestion control for multipath tcp. In *IEEE ICNP*, pages 1–10, 2012.

[6] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson. Bbr: congestion-based congestion control. *Communications of the ACM*, 60(2):58–66, 2017.

[7] M. Dong, T. Meng, D. Zarchy, E. Arslan, Y. Gilad, B. Godfrey, and M. Schapira. PCC vivace: Online-learning congestion control. In *USENIX NSDI*, pages 343–356, 2018.

[8] S. Emara, B. Li, and Y. Chen. Eagle: Refining congestion control by learning from the experts. In *IEEE INFOCOM*, pages 676–685, 2020.

[9] S. Floyd, T. Henderson, and A. Gurtov. Rfc3782: The newreno modification to tcp's fast recovery algorithm, 2004.

[10] A. Ford, C. Raiciu, M. Handley, O. Bonaventure, and C. Paasch. Rfc 6824: Tcp extensions for multipath operation with multiple addresses. *Internet Engineering Task Force*, 2013.

[11] T. Gilad, N. Rozen-Schiff, P. B. Godfrey, C. Raiciu, and M. Schapira. Mpcc: online learning multipath transport. In *ACM CoNEXT*, pages 121–135, 2020.

[12] S. Ha, I. Rhee, and L. Xu. Cubic: a new tcp-friendly high-speed tcp variant. *ACM SIGOPS operating systems review*, 42(5):64–74, 2008.

[13] J. Han, Y. Xing, K. Xue, D. S. Wei, G. Xue, and P. Hong. Leveraging coupled bbr and adaptive packet scheduling to boost mptcp. *arXiv preprint arXiv:2002.06284*, 2020.

[14] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[15] M. Honda, Y. Nishida, L. Eggert, P. Sarolahti, and H. Tokuda. Multipath congestion control for shared bottleneck. In *PFLDNeT workshop*, volume 357, page 378, 2009.

[16] J. Iyengar, P. Amer, and R. Stewart. Concurrent multipath transfer using sctp multihoming over independent end-to-end paths. *IEEE/ACM Transactions on Networking*, 14(5):951–964, 2006.

[17] R. K. Jain, D.-M. W. Chiu, W. R. Hawe, et al. A quantitative measure of fairness and discrimination. *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA*, 1984.

[18] N. Jay, N. Rotman, B. Godfrey, M. Schapira, and A. Tamar. A deep reinforcement learning perspective on internet congestion control. In *PMLR ICML*, pages 3050–3059, 2019.

[19] R. Khalili, N. Gast, M. Popovic, et al. Opportunistic linked-increases congestion control algorithm for mptcp. 2013.

[20] M. Kheirkhah, I. Wakeman, and G. Parisis. Multipath-tcp in ns-3. *arXiv preprint arXiv:1510.07721*, 2015.

[21] T. A. Le, C. S. Hong, and S. Lee. Mpcubic: An extended cubic tcp for multiple paths over high bandwidth-delay networks. In *ICTC 2011*, pages 34–39, 2011.

[22] 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 Journal on Selected Areas in Communications*, 37(11):2621–2633, 2019.

[23] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[24] Y.-s. Lim, E. M. Nahum, D. Towsley, and R. J. Gibbens. Ecf: An mptcp path scheduler to manage heterogeneous paths. In *ACM CoNEXT*, pages 147–159, 2017.

[25] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[26] C. Paasch et al. Improving multipath tcp. *Diss. Universit'e catholique de Louvain (UCL), London*, 2014.

[27] Q. Peng, A. Walid, and S. H. Low. Multipath tcp algorithms: theory and design. *ACM SIGMETRICS Performance Evaluation Review*, 41(1):305–316, 2013.

[28] B. Radunovic and J.-Y. Le Boudec. A unified framework for max-min and min-max fairness with applications. *IEEE/ACM Transactions on networking*, 15(5):1073–1083, 2007.

[29] C. Raiciu, M. Handley, and D. Wischik. Coupled congestion control for multipath transport protocols. Technical report, IETF RFC 6356, Oct, 2011.

[30] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[31] A. Walid, Q. Peng, J. Hwang, and S. Low. Balanced linked adaptation congestion control algorithm for mptcp. *Working Draft, IETF Secretariat, Internet-Draft draft-walid-mptcp-congestion-control-04*, 2016.

[32] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley. Design, implementation and evaluation of congestion control for multipath tcp. In *USENIX NSDI*, volume 11, pages 8–8, 2011.

[33] M. Xu et al. Delay-based congestion control for mptcp, draft-xu-mptcp-congestion-control-05. bd, 2017.

[34] Z. Xu, J. Tang, C. Yin, Y. Wang, and G. Xue. Experience-driven congestion control: When multi-path tcp meets deep reinforcement learning. *IEEE Journal on Selected Areas in Communications*, 37(6):1325–1336, 2019.

[35] H. Zhang, W. Li, S. Gao, X. Wang, and B. Ye. Reles: A neural adaptive multipath scheduler based on deep reinforcement learning. In *IEEE INFOCOM*, pages 1648–1656, 2019.