*future internet*

MDPI

*Article*

# DDPG-MPCC: An Experience Driven Multipath Performance Oriented Congestion Control

**Shiva Raj Pokhrel [1,*], Jonathan Kua [1], Deol Satish [1], Sebnem Ozer [2], Jeff Howe [2] and Anwar Walid [3]**

[1] IoT Research Lab, Deakin University, Geelong 3220, Australia
[2] Comcast Corporation, Philadelphia, PA 19103, USA
[3] Amazon Science, New York, NY 10001, USA
* Correspondence: shiva.pokhrel@deakin.edu.au

**Abstract:** We introduce a novel multipath data transport approach at the transport layer referred to as '*Deep Deterministic Policy Gradient for Multipath Performance-oriented Congestion Control*' (DDPG-MPCC), which leverages deep reinforcement learning to enhance congestion management in multipath networks. Our method combines DDPG with online convex optimization to optimize fairness and performance in simultaneously challenging multipath internet congestion control scenarios. Through experiments by developing kernel implementation, we show how DDPG-MPCC performs compared to the state-of-the-art solutions.

**Keywords:** multipath TCP; congestion control; experience-driven approach; performance-oriented congestion control; deep learning



**Citation:** Pokhrel, S.R.; Kua, J.; Satish, D.; Ozer, S.; Howe, J.; Walid, A. DDPG-MPCC: An Experience Driven Multipath Performance Oriented Congestion Control. *Future Internet* **2024**, *16*, 37. https://doi.org/10.3390/fi16020037

Academic Editor: Gianluigi Ferrari

Received: 24 November 2023
Revised: 23 December 2023
Accepted: 17 January 2024
Published: 23 January 2024



## 1. Introduction

New Transmission Control Protocol (TCP) designs for efficient end-to-end transport of Internet traffic, such as BBR (Bottleneck Bandwidth and Round-trip propagation time) [1], PCC (**performance oriented congestion control**) [2], PCC Vivace [3], and TCP Copa [4] do not make use of numerous available network paths and multiple interfaces. By using several paths, Multipath TCP (MPTCP) [5–7] can potentially increase the data transport rate, reduce loss and delay by redirecting traffic to less crowded paths, and improve fault tolerance by diverting traffic away from broken paths [8].

Distributing a single Internet connection traffic across multiple network interfaces, MPTCP [5–7,9] can increase device access throughput and dependability (such as smartphones and tablets) that use WiFi and 5G/6G cellular interfaces simultaneously. However, even though this scenario and many others serve as the main impetus for the development of MPTCP, recent experiences with MPTCP in simulated and real networks have shown that it often performs poorly in these scenarios. This comes as no surprise since the most popular MPTCP variants are all extensions of the seminal singlepath loss-based congestion control scheme, which has been around for several decades [10]. Almost all MPTCPs inherit TCP's well-known performance problems, particularly its inability to gracefully adjust to varying network dynamics, and poor response to link impairments like noncongestion loss, unfairness, bufferbloat, etc.

Managing congestion effectively in a multipath environment, with MPTCP, presents significant design challenges [7,11]. Like traditional TCP, MPTCP needs to adjust its transmission rates to cope with changing network conditions. However, MPTCP introduces the complexity of distributing data traffic across multiple paths to alleviate congestion, adding an extra layer of consideration for its congestion control mechanisms. Another complicating factor in the multipath congestion design is the potential for multiple subflows from the same MPTCP connection to compete with each other in network bottlenecks.

In the literature on MPTCP design, it is known that every multipath congestion control design *must meet three primary goals*: (i) the total perceived throughput of MPTCP should

be at least as good as the best TCP on any of its paths; (ii) MPTCP should make efficient use of the network by rerouting traffic away from congested paths, and (iii) MPTCP should not be more aggressive than a TCP flow when multiple subflows of an MPTCP share the same bottleneck link. Therefore, for efficient congestion control, any efficient multipath data networking protocol requires *utilizing past experience* along with *performance orientation* for optimal data transport. To this end, we need a new hybrid framework that can combine deterministic policy gradients (e.g., Deep Deterministic Policy Gradient (DDPG) for jointly exploring and exploiting past experiences [12]) with performance-oriented online optimization.

Fairness between competing TCP and MPTCP connections has been reported to be impossible by simply executing state-of-the-art TCPs (such as PCC, BBR, or Copa) over each path. When many subflows of an MPTCP connection must pass through the same bottleneck path, such an approach (executing the best TCP in each path) leads to excessive aggression, which undermines the design goals (goal iii, as mentioned earlier). Therefore, by using the online optimization approach that has been tested for singlepath TCP (PCC), we redesign MPTCP from a joint perspective of being driven by past experience and controlled by online optimization. This is because PCC design [2,3] is attractive as it requires only a few prior assumptions about the network and is known to deliver excellent performance in harsh network conditions consistently. Furthermore, Multipath PCC (MPCC) [5] without accounting for past experience has already been tested and has been shown to deliver promising results.

In this paper, we take an original step in developing a preliminary design of DDPG-MPCC, which aims to enhance the network utility maximization capabilities of MPCC in time-varying network settings. Our design assumptions are evaluated with realistic network experimental results. In other words, we present distinctive research contributions in the following ways:

1. We pioneered the development of an initial design for DDPG-MPCC, representing a novel approach to enhancing the network utility maximization capabilities of MPCC (Sections 2 and 4).
2. Our focus is specifically on addressing the challenges posed by time-varying network settings, demonstrating the relevance and adaptability of our proposed design (Sections 2.1 and 2.2).
3. The validity of our design assumptions is rigorously assessed through experimental evaluations conducted in realistic network environments, providing empirical support for the effectiveness of our approach (Sections 3 and 4).

### *1.1. Literature*

The literature on TCP congestion control design with AI and ML is very rich [13]. Of particular importance to this work is PCC [2,3], where the sender observes the connection between its actions and its experienced performance, which allows it to consistently take actions and improve performance. The case of BBR [1] is slightly different. BBR updates its rate according to the measurement of the data delivered and the Round Trip Time (RTT), while Copa [4] implements the delay-based congestion avoidance mechanism.

The design of MPTCP has included several advances in ML [5,7]. Some of them used deterministic policy gradients to discover the most effective congestion management solutions, while others suggested DRL-based designs to reduce the size of out-of-order delivery from diverse pathways. Multipath congestion management strategies [5,7,9,14] using DRL are pertinent to our study.

In MPCC [5], an online learning approach is developed for multipath data networking but inherits limitations along with the PCC control mechanisms. MPCC [5] presents an innovative extension to harness surrounding bandwidths attentively, which is discussed in detail later (Section 1.2). In addition, there are several other articles that conduct DRL-based MPTCP designs with different objectives [7,15,16]. Li et al. [15] proposed SmartCC, which observes the environment and takes certain actions to adjust the congestion window to

adapt to the varying network situations. Xu et al. [16] developed a DRL agent that leverages the emerging deterministic policy gradient to train its networks and adjust the congestion window in every action. Tianle et al. [17] implemented a deep deterministic policy gradient in satellite communications to adjust congestion rates. Jonghwan et al. [18] proposed using a machine learning model for path management in mobile devices over mptcp where signal strength and other quality metrics associated with Wi-Fi are also taken into consideration.

In this paper, *we develop a novel hybrid DDPG inside MPCC to intelligently drive packet flows across multiple paths where the DDPG module (i) maximizes the utility of the underlying network (ii) rewards an increase in throughput and (iii) ameliorates the adverse impact of latency and loss.*

### *1.2. How MPCC Works?*

MPCC aims to independently and asynchronously optimize a local utility function defined for each MPTCP subflow. For example, consider an MPTCP connection $i$ with $d$ subflows. When looking from the perspective of a subflow $j'$, the sending rate of all the other subflows except a tagged subflow $j$ can be assumed to be fixed and represented by $c_i^k$. Then, the utility function of the tagged subflow $j$ denoted by $U_i^j$ can be computed based on its own sending rate of $x_i^j$ and the experienced round trip time $RTT_j$ as [3]

$$\begin{aligned} U_i^j &= \Big(\sum_{k\neq j}^{d} c_i^k + x_i^j\Big)^{\alpha} - \beta L_j\Big(\sum_{k\neq j}^{d} c_i^k + x_i^j\Big) \\ &\quad -\gamma\Big(\sum_{k\neq j}^{d} c_i^k + x_i^j\Big)\frac{dRTT_j}{dT} \end{aligned} \tag{1}$$

where $L_j$ is the loss rate and $\frac{dRTT_j}{dT}$ quantifies the delay variation observed by the subflow $j$, and $0 \leq \alpha < 1$, $\beta > 3$ and $\gamma \geq 0$ are the parameters for balancing performance objectives. However, from a higher MPTCP connection level perspective, the utility function of the MPTCP $i$ can be estimated as

$$U_i = \Big(\sum_{k}^{d} x_i^k\Big)^{\alpha} - \max_{k\in d}\{\beta L_k + \gamma\frac{dRTT_k}{dT}\}\sum_{k}^{d} x_i^k \tag{2}$$

It has been reported that [5], with MPCC we can achieve better performance by decentralizing optimization across subflows. This can be clearly observed in (1) as the utility function or the utility that will lead to rate changes is dependent on its own locally perceived success, loss, and delay statistics. MPCC adopts the concept of **Monitoring Intervals** (MI), which is the duration of a time long enough to gather sufficient statistics for each of the MPTCP subflows. First, MPCC selects a sending rate for the subflow for each MI, and then it computes the utility value of the subflows (after the defined MI interval to gather the required statistics). In this process, the subflow transitions occur in three different states, namely, *slow-start*, *probing* and *moving* [2,3,5]:

(i) In the *slow start phase*, the sending rate is doubled in every MI until the utility decreases. MPCC can utilize the previous experience of the sending rates to begin the probe.

(ii) In the *probe phase*, MPCC probes the subflow with a higher sending rate $x + \omega$ and a lower sending rate of $x - \omega$, where $\omega$ is the fraction of the total sending rate of all subflows within the MPTCP connection (that is, $\omega = x_i^k / \sum x_i^k$). It is worth noting that the probing phase helps to decide the direction in which the rate should be modified, which directs the beginning of the moving phase.

(iii) In the *moving phase*, MPCC estimates the gradient of the utility function by using the utilities and the rates from the two preceding MIs. With the computed gradient, MPCC mandates the rate change or the step increase/decrease in the direction towards the increasing pattern for the utility of the connection. It can be seen that the

MPCC inherited such specific and celebrated mechanisms (e.g., rate amplification and changed bounds) from PCC Vivace [3] to seamlessly adjust the speed of data transport for the Internet. For example, if the estimated utility decreases, then it can quickly switch back to the probing phase.

## 2. Experience Driven MPCC Design

We first formulate a Markov Decision Process (MDP) to model the learning process of the proposed DDPG-MPCC. MDP formalizes the environment for DDPG learning. The learning process starts by observing the state of the environment $s^j(t)$ from the MPTCP connection of the subflow $j$ and decides by taking an action according to the policy $\pi(a \mid s)$. The action of the DDPG agent changes the environment state to $s^j(t+1)$, and the model receives a reward $R(t)$. The state of the environment is represented by the tuple: sending rate $x^j(t)$, round trip time $RTT^j(t)$, loss ratio $L^j(t)$ of each MPTCP subflow. Actions are represented by sending rate adjustments (increments/decrements) over an MPTCP subflow.

The proposed DDPG agent consists of three main components, (i) State space: $s_t = \left[x^j(t), RTT^j(t), L^j(t)\right]$ (ii) Action space: $a_t = \left[x_1, x_2, \ldots, x^j\right]$; and (iii) Reward: for simplicity, reward $R = U^j$ (given by (1)). For simplicity, we use the MPCC utility as a reward after each MI.

DDPG is explicitly adapted for continuous action spaces and therefore has the potential to transform the MPCC. In our design of DDPG-MPCC, we have flexibly coupled a deterministic policy network (*Actor*) and Q-network (*Critic*). The underlying idea is to capture the network dynamics inherently by using a policy iterating mechanism, which is applied to alternate between policy improvement (actor) and policy evaluation (critic). It is worth noting that the actor is meant to improve the policy using a policy gradient, while the critic evaluates the policy for the current parameters.

DDPG allows the implementation of an experienced pool and two target networks to improve stability and convergence. For higher performance and better convergence, we adopt replay buffers, where a set of previous experiences is stored. It contains a wide array of experiences, though it may not be suitable to keep everything as using too much experience may decelerate the learning process. Due to time-varying network dynamics, the target function fluctuates, and the training needs to cope with such fluctuations. Therefore, we adopt the target network that fixes the parameters of the target function for a fixed number of epochs and then replaces them with a newly updated target network.

We formulate an actor and critic model by representing policy function and Q-network parameters as:

$$policy \begin{cases} online : \mu(s|\theta^{\mu})\ gradient\ update\ \theta^{\mu} \\ target : \mu\left(s|\theta^{\mu'}\right)\ soft\ update\ \theta^{\mu'} \end{cases}$$

$$Q\ network \begin{cases} online : Q\left(s,a|\theta^{Q}\right)\ gradient\ update\ \theta^{Q} \\ target : Q\left(s,a|\theta^{Q'}\right)\ soft\ update\ \theta^{Q'} \end{cases}$$

The policy objective function, $J(\theta^{\mu})$, for the actor-network is computed as a weighted cumulative expected reward from all subflows:

$$J(\theta^{\mu}) = \mathbb{E}_{\theta^{\mu}}\left[R_1 + \gamma R_2 + \gamma^2 R_3 + \ldots\right]$$

with $\gamma$ exponentially decreasing the future rewards' weight.

With relevant insights from [19], we assume that the change in the policy gradient with respect to $\theta^{\mu}$ is approximately equal to the considered gradient of the Q-valued function. Therefore, the policy gradient of $J(\theta^{\mu})$ can be estimated by computing the expectation

$$\frac{\partial J(\theta^{\mu})}{\partial \theta^{\mu}} = \mathbb{E}_s\left[\frac{\partial Q\left(s,a|\theta^{Q}\right)}{\partial a}\frac{\partial \mu(s|\theta^{\mu})}{\partial \theta^{\mu}}\right]$$

Similarly, the critic network receives the action and state spaces and computes $Q(s,a)$ for evaluating the impacts of taking action $a$. It updates its parameters on the basis of the target value. The underlying gradients can be estimated as

$$\frac{\partial L(\theta^Q)}{\partial \theta^Q} = \mathbb{E}_{s,a,s'} \frac{(T_Q - Q(s,a|\theta^Q)\partial Q(s,a|\theta^Q)}{\partial \theta^Q} \tag{3}$$

where,

$$T_Q = r + \gamma Q'\left(s', \mu(s'|\theta^{\mu'})|\theta^{Q'}\right)$$

Using Polyak averaging after each MIs, the parameters of the target network are updated from the parameters of the online network, i.e., the soft update of the target policy and Q network can be estimated as: $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$; $\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'}$. The Algorithm 1 demonstrates detailed steps of the DDPG-MPCC algorithm devised.

**Algorithm 1** Pseudocode of DDPG-MPCC Algorithm

Initialization of DDPG Params: critic $Q(s,a|\theta^Q)$, actor $\mu(s|\theta^{\mu})$ Replay Buffer B and $Q\left(s,a|\theta^{Q'}\right)$, $\mu\left(s|\theta^{\mu'}\right)$
**while** $Episode < Max$ **do**
  **for** $t = 1, 2, \ldots$ MI **do**
    Run Ornstein–Uhlenbeck Process for exploration
    Determine $a_t$ using policy network and exploration
    Apply action $a_t$ Compute system reward $R_t$
    Determine next state $s_{t+1}$
    Save transitions $(s_t, a_t, R_t, s_{t+1})$ in Replay B
  **end for**
  Retrieve $\mathcal{N}$ transition from B $(s_i, a_i, R_i, s_{i+1})$
  $\mathrm{Y_i} = \mathrm{R_i} + \gamma Q'(s_{\mathrm{i+1}}, \mu'(s_{\mathrm{i+1}}|\theta^{\mu'})|\theta^{Q'})$
  Renew Critic: min $\left(L = 1/N \sum_i (y_{\mathrm{i}} - Q(s_{\mathrm{i}}, a_{\mathrm{i}}|\theta^Q))^2\right)$
  Renew Actor Policy using updated policy gradient
  Renew Target Networks (using hyperparameter, $\tau$):

$$\begin{cases} \theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'} \\ \theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'} \end{cases}$$

**end while**

### *2.1. Key Components of DDPG-MPCC: Engineering Interpretation*

DDPG-MPCC involves several key components, each contributing to its functionality. In the following, as shown in Figure 1, we discuss these components along with a conceptual diagram.

*State Space.* The state space represents the current state of the system, capturing relevant information for decision making. It includes details such as network conditions, available paths, congestion levels, and historical data.

*Action Space.* The action space consists of the possible actions the agent can take in a given state. Actions involve selecting the combination of paths for data transmission, adjusting sending rates, and managing congestion control.

*Reward Mechanism.* The reward mechanism provides feedback to the agent based on its actions, guiding it toward desirable outcomes. Rewards can be derived from achieving efficient data transmission, minimizing congestion, or optimizing network utility.

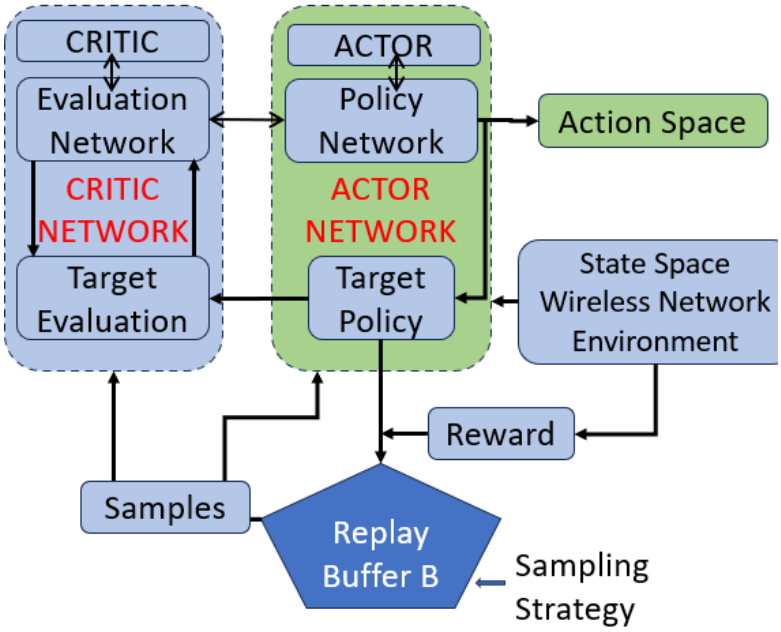


**Figure 1.** Abstract view of the interrelationship between DDPG-MPCC Modules.

*Actor Network.* The actor-network determines the policy or strategy for selecting actions in a given state. In DDPG-MPCC, the actor-network decides the optimal path combination and sending rates based on the current state.

*Critic Network.* The critic network of DDPG-MPCC evaluates the actions taken by the actor, providing feedback on their effectiveness, which assesses the selected actions' impact on overall system performance. ctor.

*Experience Replay Buffer.* The experience replay buffer B stores past experiences to break the temporal correlation in sequential data. It retains historical data to improve the stability and efficiency of the learning process.

*Training Process.* The training process involves iteratively updating the actor and critic networks based on experiences and rewards. It refines the policy and value functions to improve decision-making over time.

### *2.2. DDPG-MPCC Implementation*

Our implementation is tailored so that DDPG-MPCC retains the sound characteristics of MPCC while the DDPG plugin module accelerates the learning and bandwidth exploration process. MPCC starts with the slow start phase, multiplicatively increasing its sending rate until the utility decreases. When the utility decreases, it will go to the probing phase, exploring a network with a higher and lower sending rate. This probing mechanism helps congestion control to decide the direction in which the rate should be changed. In the moving phase, DDPG-MPCC will determine how much the rate must change to increase the utility. When utility decreases, it falls back to the probing phase. Our project page is available at github https://github.com/MPTCP-FreeBSD (accessed on 1 January 2024), and the implementation of this paper will be publicly available at DDPG-MPCC repository https://github.com/MPTCP-FreeBSD/DDPG-MPCC-SRC (accessed on 1 January 2024). The implementation and experiment of DDPG-MPCC have been fully conducted by the author, D. Satish, as a part of his honors thesis for SIT723, School of IT, Deakin University.

The initial data were collected through the kernel logs while setting up the network scenario over CloudLab. The CloudLab clusters consist of nearly one thousand machines distributed across three sites in the United States: Utah, Wisconsin, and South Carolina. These machines are interoperable with existing testbeds, such as GENI and Emulab, allowing us to utilize hardware at dozens of sites around the world. See https://www.cloudlab.us/ for details.

We train the DDPG model offline using the collected logs. We can also change the utility function to prioritize lower latency and loss rather than high utilization or vice versa. In our experiments, each episode starts by resetting the whole environment, and the training session always begins with a new data point whose utility is between 40% and 70%. Each episode trains the underlying agent to capture the interrelationship between action, states, and reward. We execute 1000 actions or steps, improving the entire episode's reward. This is the training of the model based on the data-networking dynamics collected over the MPCC environment data for 1000 episodes. In actual deployment, the model will be used at specific intervals to receive data from the kernel space and calculate the best action given the current state of the network.

### *2.3. Testbed Setup and Normalization*

We implemented the MPCC kernel and incorporated an offline DDPG module for training and testing using realistic datasets. In contrast to the default MPTCP scheduler, our rate-based approach in DDPG-MPCC signals the scheduler to set a high congestion window, unless unforeseen consequences occur. To maintain tractability, following [5], we adopted the assumption in a rate-based multipath scheduler that a subflow becomes unavailable when 10% of the packets needed to sustain the current sending rate for the current RTT duration are queued for transmission in the path. The DDPG-MPCC scheduler and kernel design extend the work in [5] by importing logs and integrating the DDPG module into our Linux client node. The extraction of kernel logs provides the necessary data for the training and offline learning of DDPG-MPCC.

Figure 2 illustrates the network topology and the three scenarios employed in evaluating the multipath protocols. Our testbed, created over CloudLab, features a typical client and server connected through two emulators and two routers simulating WiFi and 5G paths. This test bench comprises six nodes, including a client, server, emulators, and routers, as depicted in Figure 3.

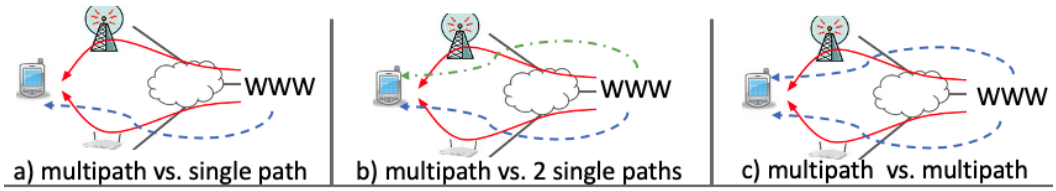


**Figure 2.** Three scenarios to quantify the performance of MPCC across varying bandwidth: (**a**) MPTCP connection vs. singlepath connection; (**b**) MPTCP connection vs. two singlepath connections; (**c**) Two MPTCP connections contending with each other in both paths.

While testing and comparing various congestion controls, we vary the bandwidth of the route by changing the bandwidth of the router's interface. Similarly, we insert a certain loss percentage in our emulator nodes by using the Linux traffic control or utility for the interface in that route. We run five 300s iperf3 tests in multiple network scenarios for varying bandwidth and varying loss percentages where we keep the other environment variables to default.

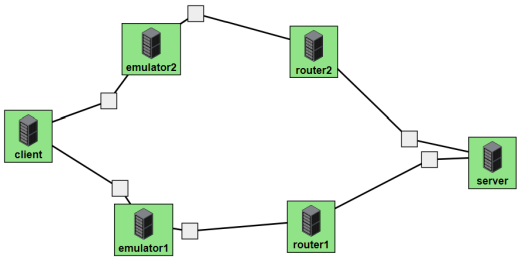


**Figure 3.** Cloulab Testbed Setup with Two paths.

## 3. Performance Evaluation

While testing and comparing various congestion controls, we vary the bandwidth of the route by changing the bandwidth of the router interface. Similarly, we insert a certain percentage of loss into our emulator nodes by using the Linux traffic control for the interface on that route. We run several *iPerf3* tests in network scenarios shown in Figure 2 by varying the bandwidth and varying the percentages of losses without changing other network parameters (along the lines of [5]).

Unless otherwise specified, all path latencies, bandwidths, and buffer capacities are 40ms, 100Mbps, and 400 packets, respectively. Along the lines of MPCC [5], in Figures 4 and 5, we compare MPCC with the Linux kernel-implemented MPTCP versions Lia, Olia, and Balia, as well as with the use of singlepath TCP (Reno and BBR) for each subflow. The details about Linux kernel-implemented MPTCP versions (Lia, Olia, and Balia) and TCP (Reno, Cubic, and BBR) are explained in [5,7] and therefore we include essential details only.

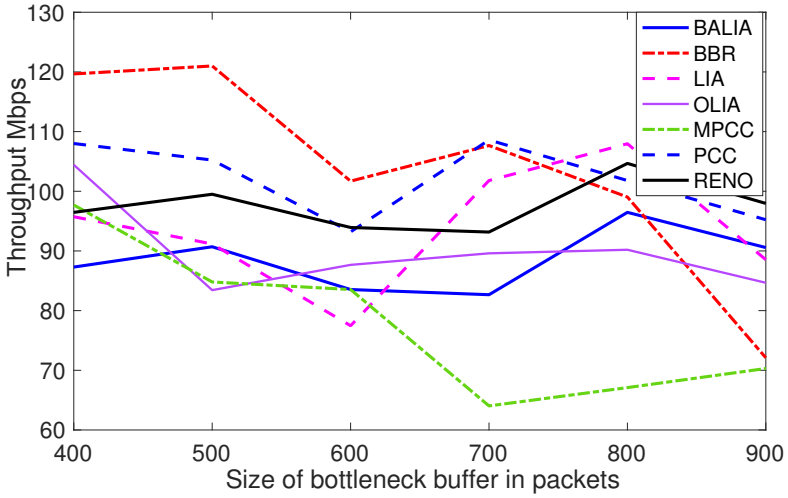


**Figure 4.** Throughput comparison of one of the shared paths of MPCC with the state-of-the-art MPTCP and TCP protocols over the variation in the size of bottleneck buffer in the path (network scenario a).

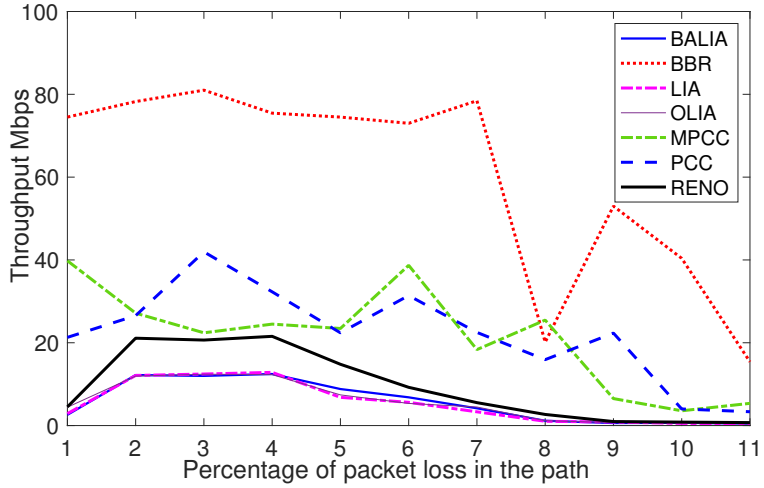


**Figure 5.** Throughput comparison of MPCC with the state-of-the-art MPTCP and TCP protocols over the variation in the percentage of path loss (network scenario b).

We repeat the experiment outlined in Network Scenario a, the multipath sender competes with a single path sender (PCC vs. MPCC, TCP Reno vs. MPTCP). As shown

in Figure 4, when a multipath protocol fails to exploit path 1 effectively, it will become more aggressive on path 2, resulting in a decrease in good performance for the single path connection.

As shown in Figure 5, MPTCP responds to packet loss by scaling down the sending rate multiplicatively. In sharp contrast, MPCC maintains a higher throughput across the full range and only drops performance when the loss rate is several times greater than that, which has the equivalent effect on MPTCP. These observations have been partly reported in [5], all of which have been extensively experimented with, and the logs are gathered to generate the data sets required to train the DDPG module.

By implementing the three different scenarios shown in Figure 2, the internet traffic and performance parameters are captured through custom logging functions placed within the kernel to ensure accurate data analysis. This is to prepare sufficient data for offline training and setting up the input environment of the DDPG module. For simplicity, considering the data distribution of the inputs for DDPG training, we normalized values (between 0 and 1) to simplify the learning process. Based on our observations across several experimental runs, the performance logs have drastic variations, and normalization allows us to eliminate such skewed data. It is known in the DDPG literature that such an approach helps to improve the convergence of training dynamics and achieves better stability.

After gathering training data from the kernel logs, we train our DDPG-MPCC using the three networks and several runs. In the offline training procedure, we ensure that each training episode begins with a new data point and always converges. We then tested our trained model for 100 MIs using the test dataset obtained in the same fashion when DDPG-MPCC competes with MPCC across the network in scenario c) under the same parameter settings. In each MI, we calculate the tuples (average sending rates, average utility, and average reward) and plot them to examine how the sending rate and utility of DDPG-MPCC and MPCC improve with MIs. Figures 6–8 illustrate the results.

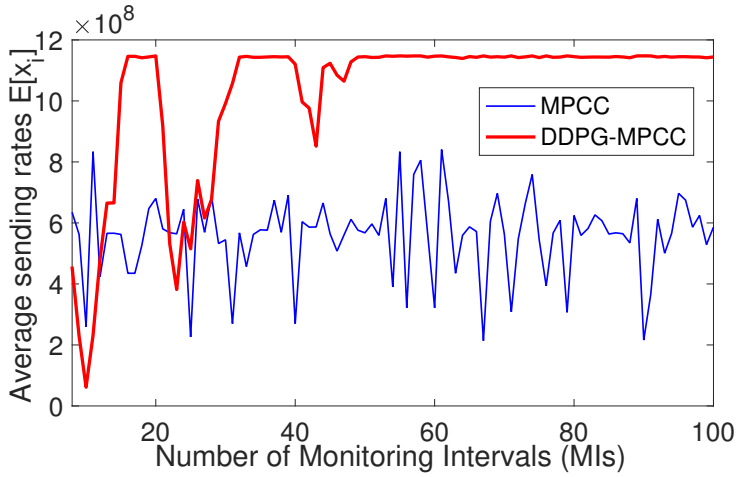


**Figure 6.** Comparison of the evolution of the average sending rates of MPCC [5] and DDPG-MPCC on the number of MI (network scenario c, Figure 2).

We can see in Figure 6 that the average sending rate of DDPG-MPCC is very low at the start, with a bit of contention with MPCC for some initial MIs. It demonstrates substantial increments over higher MIs and attains the maximum without any visible (adverse) impacts on the dynamics and performance of the competing MPCC rates.

In Figure 7, we present our new observations of the evolutionary graphs of the utilities experienced by DDPG-MPCC and MPCC in relation to the number of MIs. We clearly observe that the utility of the DDPG-MPCC, which is initially rather low, improves significantly more than the utility of the MPCC. Observe that the utility of DDPG-MPCC increases as MIs progress and follows the same trend as that of the sending rates in Figure 6. To further explore and understand such an important capability of maximizing

utility with DDPG-MPCC, we quantify the average rewards over MIs, and we can see in Figure 8 that the performance of DDPG-MPCC in terms of utility maximization is due to the intrinsic ability to gradually move towards improving the rewards. It is noted that the average episodic reward for MI in Figure 8 increases sharply, thus maximizing the utility of the network.

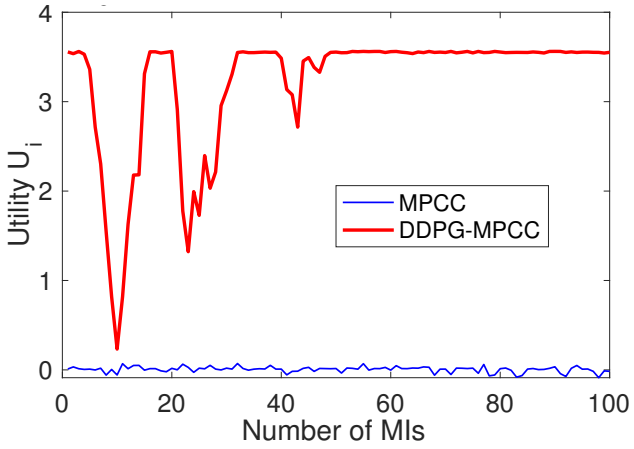


**Figure 7.** Evolution of the average utilities of MPCC [5] and DDPG-MPCC over the number of MI (network scenario c, compared to Figure 6).

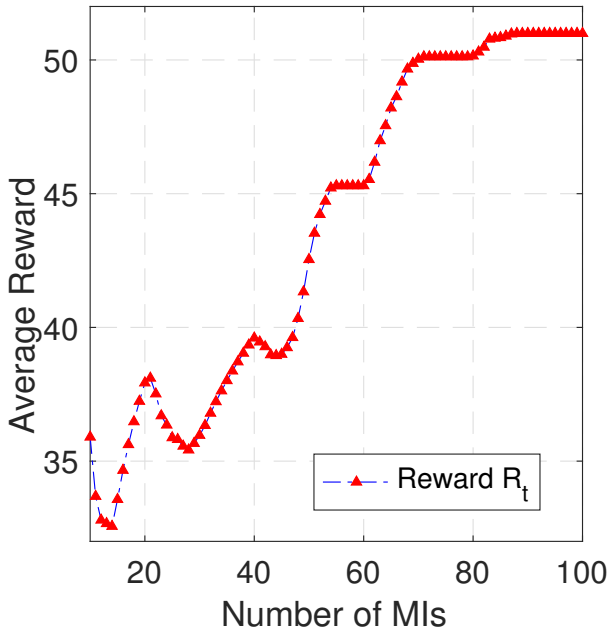


**Figure 8.** Evolution of the average reward ($R_t$) of the trained DDPG over the number of MIs, while DDPG-MPCC and MPCC for utility maximization as demonstrated in Figure 6.

## 4. Discussion and Directions

The proposed DDPG-MPCC, designed to boost MPCC performance in dynamic network scenarios, integrates a deterministic policy network and Q-network. This reinforcement learning agent demonstrates superiority over MPCC, excelling in adaptive learning, handling complex state spaces, providing enhanced stability, balancing exploration-exploitation, and optimizing policies. In this section, we will discuss how our experimental results reveal improved utility and reward over MIs, showcasing its potential for dynamic network optimization. However, effective real-world implementation requires careful consideration of reinforcement learning parameters and extensive further research with thorough evaluations.

### *4.1. DDPG-MPCC Action Is Optimal*

DDPG-MPCC is designed to enhance the performance of MPCC, particularly in the context of time-varying network scenarios. Recall that the proposed DDPG agent is composed of three main components: (i) State space, (ii) Action space, and (iii) Reward. For tractability [7], we use the MPCC utility as a reward after each Mutual Information (MI). In Figure 6, we illustrate the comparison of the evolution of the average sending rates (actions) between MPCC and DDPG-MPCC over the number of MIs. DDPG is specifically tailored for continuous action spaces, providing an opportunity to enhance the MPCC.

In our DDPG-MPCC design, we seamlessly integrate a deterministic policy network (Actor) and a Q-network (Critic). The core concept involves capturing network dynamics through a policy iteration mechanism that alternates between policy improvement (actor) and policy evaluation (critic). This process is depicted in Figure 7, showcasing the evolution of average utilities (shaping optimal policy) for MPCC [5] and DDPG-MPCC over the number of MIs.

The actor's role is to enhance the policy using a policy gradient, while the critic evaluates the policy based on current parameters. We establish an actor and critic model by representing the policy function and Q-network parameters. The policy objective function, $J(\theta_\mu)$, for the actor-network is calculated as a weighted cumulative expected reward. Additionally, Figure 8 illustrates the evolution of the average reward ($R_t$) of the trained DDPG, that is, the optimal policy over the number of MIs, while DDPG-MPCC and MPCC aim at maximizing utility.

### *4.2. Perfromance Evaluation with MPTCPs and Data Collection*

In contrast to MPTCP, which responds to packet loss by scaling down the sending rate multiplicatively, our proposed DDPG-MPCC demonstrates a novel approach. It is known that MPCC [5] maintains a higher throughput across a broad range of loss rates and only experiences performance degradation when the loss rate significantly surpasses that of MPTCP. These distinctive observations have been rigorously tested and analyzed; important findings are shown in Figures 4 and 5. When a multipath protocol fails to effectively utilize path 1, it tends to become more aggressive on path 2. This behavior leads to a decrease in the overall performance of the singlepath connection. The insights gained from the experiments have been instrumental in collecting logs to generate datasets for training the innovative DDPG module developed in this research work.

We collect Internet traffic and performance parameters by implementing three scenarios (Figure 2) using custom log functions within the kernel. These data are crucial for offline training and configuration of the input environment of the DDPG module. Normalizing values between 0 and 1 simplifies the learning process, addresses significant variations in performance logs, and contributes to improved convergence and stability in the dynamics of the DDPG training. In our results earlier, we clearly illustrate our new findings on evolutionary graphs that depict the utility of DDPG-MPCC and MPCC in relation to the number of MIs. In particular, DDPG-MPCC exhibits a substantial improvement in utility compared to MPCC, and this improvement aligns with the trend observed in the sending rates illustrated in Figure 6. To dig deeper into the utility maximization capability of DDPG-MPCC, we quantify the average rewards over MIs (Figure 8).

### *4.3. Superiority of DDPG-MPCC over MPCC*

The superiority of DDPG-MPCC over MPCC can be attributed to its enhanced learning and adaptive capabilities. DDPG introduces a Deep Deterministic Policy Gradient-based reinforcement learning framework that enables the agent, referred to as *DDPG-MPCC*, to learn and improve its policy over time through interactions with the environment.

Here are some reasons why DDPG-MPCC may outperform MPCC:

- *Adaptive Learning:* DDPG-MPCC adapts its policies based on continuous feedback from the environment. This adaptability allows it to dynamically adjust to varying network conditions, optimizing performance in real time.

- *Handling Complex State Spaces:* DDPG is well-suited for problems with continuous action spaces, making it effective in scenarios where MPCC faces challenges. The ability to handle complex state spaces allows DDPG-MPCC to navigate a broader range of network conditions.
- *Enhanced Stability:* The reinforcement learning framework of DDPG often contributes to stable and consistent learning over time. This stability can lead to more reliable and robust performance compared to non-learning or less adaptive approaches.
- *Exploration and Exploitation:* DDPG-MPCC employs a balance between exploration and exploitation, exploring new strategies while exploiting known effective ones. This can lead to more effective decision-making in diverse and dynamic network scenarios.
- *Improved Policy Optimization:* DDPG-MPCC uses a policy gradient approach to optimize its policies. This methodology can lead to more refined and effective policies compared to traditional methods, potentially resulting in improved utility and performance.

DDPG is a popular deep reinforcement learning algorithm applied to continuous control problems like congestion control, but it has its drawbacks. It can become unstable, highly dependent on searching for optimal hyperparameters, and is susceptible to overestimating Q values in the critic network. This overestimation can lead to the agent being trapped in local optima or suffering from disastrous forgetting over time. Twin-delayed DDPG addresses the overestimation bias but may not fully exploit performance due to underestimation bias. Authors in [20] introduce Twin Average Delayed DDPG, tailored to TD3, and demonstrate superior performance compared to TD3 in challenging continuous control environments, which requires further investigation to be adopted for the proposed DDPG-MPCC in the future.

It is important to note that the effectiveness of DDPG-MPCC depends on various factors, including the design of the reinforcement learning setup, the choice of hyperparameters, and the specific characteristics of the network environment. Experimental validation and a thorough performance evaluation, as detailed in this paper, are typically essential to affirm the benefits of DDPG-MPCC over MPCC in a given context, which requires further extensive evaluations *in the wild*.

### *4.4. Potential Future Direction*

Future research in MPCC with ML could explore integrated approaches that jointly optimize congestion control and packet scheduling. Current works often address these aspects separately, but a holistic view, combining both elements, holds promise for more efficient and adaptive MPCC systems. This direction could lead to the development of models that dynamically adjust MPCC parameters in response to changing network conditions, enhancing overall performance.

A novel avenue involves leveraging predictive analytics into MPCC for anticipatory path selection. ML algorithms could analyze historical data to predict changes in path characteristics, enabling MPCC to make proactive decisions for optimal performance. This forward-looking approach aims to enhance decision-making in MPCC by leveraging predictive insights, potentially mitigating performance issues before they occur.

The future of MPCC research with ML also encompasses enhancing security, improving explainability, and conducting extensive real-world evaluations. Researchers can explore ML applications for anomaly detection, adaptive security measures, and adversarial training to fortify MPCC against potential threats. Additionally, incorporating explainable AI features ensures transparency in MPCC decision-making. Real-world deployments and evaluations will be crucial to validate the effectiveness, scalability, and robustness of ML-based enhancements in diverse network environments.

## 5. State of the Art of Multipath Congestion Control

In recent years, significant strides have been made in optimizing MPTCP performance through the development of packet scheduling and congestion control mechanisms [21–24]. Noteworthy contributions include Ji et al.'s adaptive approach for multipath live stream-

ing [25] and the shared bottleneck multipath detection mechanism proposed by [26]. Multipath congestion control has been addressed in works such as [27–30], while advancements in packet scheduling are discussed in [31–33]. The MPTCP scheduler, employing the "min–RTT" policy [34], allocates packets to the path with the smallest RTT and has been widely studied [15,16,28,35–38].

Recent machine learning advances have influenced MPTCP redesign, with notable contributions focusing on control strategies [15–18,39,40]. Mai et al. [17] applied a deterministic policy gradient for efficient control strategies, while Liao et al. [39] proposed multipath scheduling based on Deep Reinforcement Learning (DRL) on heterogeneous paths. Silva et al. [40] introduced an adaptive virtual reality with content-aware prioritization to enhance MPTCP's performance.

Three new multipath congestion control designs [15,16] focus on congestion control but do not address packet scheduling. Notably, all the mentioned MPTCP designs [15–18,39,40] consider congestion control or packet scheduling separately instead of jointly.

In the realm of MPTCP controllers,[16] demonstrates advantages when applied to a group of sources, diverging from the more common one-controller-per-source scenario. However, despite stability being a key goal in the original MPTCP design philosophy [6], the stability of the proposed algorithms [15–18,39,40] remains unaddressed.

## 6. Conclusions

We developed a DDPG-MPCC approach to enhance MPCC network utility maximization capabilities under time-varying network scenarios. Our initial design and evaluation of DDPG-MPCC hope to inspire more research into its theoretical and empirical assurances and how the design might be enhanced. In particular, specific observations raise the challenge of obtaining verifiable warrants extending beyond harnessing multipath networks. Some impending aspects include inadequate performance over network links with vastly varying available capacities and inefficient flow completion duration for short-lived flows, which requires further work.

**Author Contributions:** Conceptualization, S.R.P; methodology, S.R.P. and J.K.; software, D.S.; validation, D.S.; formal analysis, D.S.; investigation, S.R.P. and J.K.; resources, S.R.P. and J.K.; data curation, D.S.; writing—original draft preparation, S.R.P., J.K. and D.S.; writing—review and editing, S.R.P., J.K., S.O., J.H. and A.W.; visualization, D.S.; supervision, S.R.P., J.K., S.O., J.H. and A.W.; project administration, S.R.P. and J.K.; funding acquisition, S.R.P., J.K., S.O., J.H. and A.W. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by COMCAST Innovation Fund 2022.

**Data Availability Statement:** The data presented in this study are available at https://github.com/MPTCP-FreeBSD/DDPG-MPCC-SRC and further resources will be made available on request from the corresponding author.

**Conflicts of Interest:** Anwar Walid is an Amazon applied science manager and Jeff Howe is the Director of DOCSIS Engineering Comcast Corporation. Author S. Ozer was employed by COMCAST Corporation. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

## References

1. Cardwell, N.; Cheng, Y.; Gunn, C.S.; Yeganeh, S.H.; Jacobson, V. BBR: Congestion-based congestion control. *Commun. ACM* **2017**, *60*, 58–66. [CrossRef]
2. Dong, M.; Li, Q.; Zarchy, D.; Godfrey, P.B.; Schapira, M. PCC: Re-architecting congestion control for consistent high performance. In Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15), Oakland, CA, USA, 4–6 May 2015; pp. 395–408.
3. Dong, M.; Meng, T.; Zarchy, D.; Arslan, E.; Gilad, Y.; Godfrey, B.; Schapira, M. PCC Vivace: Online-Learning Congestion Control. In Proceedings of the 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18), Renton, WA, USA, 9–11 April 2018; pp. 343–356.
4. Arun, V.; Balakrishnan, H. Copa: Practical Delay-Based Congestion Control for the Internet. In Proceedings of the 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18), Renton, WA, USA, 9–11 April 2018; pp. 329–342.

5. Gilad, T.; Rozen-Schiff, N.; Godfrey, P.B.; Raiciu, C.; Schapira, M. MPCC: Online learning multipath transport. In Proceedings of the 16th International Conference on emerging Networking EXperiments and Technologies, Barcelona, Spain, 1–4 December 2020; pp. 121–135.
6. Ford, A.; Raiciu, C.; Handley, M.J.; Bonaventure, O. *TCP Extensions for Multipath Operation with Multiple Addresses*; RFC 6824; IETF: Wilmington, DE, USA, 2013. [CrossRef]
7. Pokhrel, S.R.; Walid, A. Learning to Harness Bandwidth with Multipath Congestion Control and Scheduling. *IEEE Trans. Mob. Comput.* **2023**, *22*, 996–1009. [CrossRef]
8. Pokhrel, S.R.; Jin, J.; Vu, H.L. Mobility-Aware Multipath Communication for Unmanned Aerial Surveillance Systems. *IEEE Trans. Veh. Technol.* **2019**, *68*, 6088–6098. [CrossRef]
9. Tian, H.; Liao, X.; Zeng, C.; Sun, D.; Zhang, J.; Chen, K. Efficient DRL-Based Congestion Control with Ultra-Low Overhead. *IEEE/ACM Trans. Netw.* **2023**. *accepted for publication*. [CrossRef]
10. Liu, J.; Huang, J.; Jiang, W.; Li, Z.; Li, Y.; Lyu, W.; Jiang, W.; Zhang, J.; Wang, J. End-to-End Congestion Control to Provide Deterministic Latency Over Internet. *IEEE Commun. Lett.* **2022**, *26*, 843–847. [CrossRef]
11. Pokhrel, S.R.; Pan, L.; Kumar, N.; Doss, R.; Vu, H.L. Multipath TCP Meets Transfer Learning: A Novel Edge-Based Learning for Industrial IoT. *IEEE Internet Things J.* **2021**, *8*, 10299–10307. [CrossRef]
12. Pan, L.; Cai, Q.; Huang, L. Softmax deep double deterministic policy gradients. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 11767–11777.
13. Quan, W.; Xu, Z.; Liu, M.; Cheng, N.; Liu, G.; Gao, D.; Zhang, H.; Shen, X.; Zhuang, W. AI-driven Packet Forwarding with Programmable Data Plane: A Survey. *IEEE Commun. Surv. Tutorials* **2022**, *25*, 762–790. [CrossRef]
14. Wu, H.; Alay, O.; Brunstrom, A.; Caso, G.; Ferlin, S. Falcon: Fast and accurate multipath scheduling using offline and online learning. *arXiv* **2022**, arXiv:2201.08969.
15. Li, W.; Zhang, H.; Gao, S.; Xue, C.; Wang, X.; Lu, S. SmartCC: A Reinforcement Learning Approach for Multipath TCP Congestion Control in Heterogeneous Networks. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 2621–2633. [CrossRef]
16. Xu, Z.; Tang, J.; Yin, C.; Wang, Y.; Xue, G. Experience-driven Congestion Control: When Multi-Path TCP Meets Deep Reinforcement Learning. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 1325–1336. [CrossRef]
17. Mai, T.; Yao, H.; Jing, Y.; Xu, X.; Wang, X.; Ji, Z. Self-learning Congestion Control of MPTCP in Satellites Communications. In Proceedings of the 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC), Tangier, Morocco, 24–28 June 2019; pp. 775–780.
18. Chung, J.; Han, D.; Kim, J.; Kim, C.K. Machine Learning Based Path Management for Mobile Devices Over MPTCP. In Proceedings of the 2017 IEEE International Conference on Big Data and Smart Computing (BigComp), Jeju, Republic of Korea, 13–16 February 2017; pp. 206–209.
19. Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; Riedmiller, M. Deterministic Policy Gradient Algorithms. In Proceedings of the 31st International Conference on Machine Learning, Beijing, China, 22–24 June 2014.
20. Tiong, T.; Saad, I.; Teo, K.T.K.; Lago, H.B. Deep Reinforcement Learning with Robust Deep Deterministic Policy Gradient. In Proceedings of the 2020 2nd International Conference on Electrical, Control and Instrumentation Engineering (ICECIE), Kuala Lumpur, Malaysia, 28 November 2020; pp. 1–5. [CrossRef]
21. Han, J.; Xue, K.; Li, J.; Zhuang, R.; Li, R.; Yu, R.; Xue, G.; Sun, Q. EdAR: An Experience-Driven Multipath Scheduler for Seamless Handoff in Mobile Networks. *IEEE Trans. Wirel. Commun.* **2023**, *22*, 6839–6852. [CrossRef]
22. Aggarwal, S.; Saha, S.K.; Khan, I.; Pathak, R.; Koutsonikolas, D.; Widmer, J. MuSher: An Agile Multipath-TCP Scheduler for Dual-Band 802.11ad/ac Wireless LANs. *IEEE/ACM Trans. Netw.* **2022**, *30*, 1879–1894. [CrossRef]
23. Wang, C.; Wang, H.; Qian, F.; Zheng, K.; Wang, C.; Mao, F.; Guo, X.; Xu, C. Experience: A Three-Year Retrospective of Large-Scale Multipath Transport Deployment for Mobile Applications. In Proceedings of the 29th Annual International Conference on Mobile Computing and Networking, Madrid, Spain, 2–6 October 2023; Association for Computing Machinery: New York, NY, USA, 2023.
24. Song, C.; Han, B.; Ji, X.; Li, Y.; Su, J. AI-driven Multipath Transmission: Empowering UAV-based Live Streaming. *IEEE Netw.* **2023**. *accepted for publication*. [CrossRef]
25. Ji, X.; Han, B.; Xu, C.; Song, C.; Su, J. Adaptive QoS-aware multipath congestion control for live streaming. *Comput. Netw.* **2023**, *220*, 109470. [CrossRef]
26. Dong, E.; Gao, P.; Yang, Y.; Xu, M.; Fu, X.; Yang, J. SmartSBD: Smart shared bottleneck detection for efficient multipath congestion control over heterogeneous networks. *Comput. Netw.* **2023**, *237*, 110047. [CrossRef]
27. Peng, Q.; Walid, A.; Hwang, J.; Low, S.H. Multipath TCP: Analysis, Design, and Implementation. *IEEE/ACM Trans. Netw.* **2016**, *24*, 596–609. [CrossRef]
28. Chiariotti, F.; Kucera, S.; Zanella, A.; Claussen, H. Analysis and Design of a Latency Control Protocol for Multi-Path Data Delivery With Pre-Defined QoS Guarantees. *IEEE/ACM Trans. Netw.* **2019**, *27*, 1165–1178. [CrossRef]
29. Khalili, R.; Gast, N.; Popovic, M.; Le Boudec, J.Y. MPTCP is Not Pareto-optimal: Performance Issues and a Possible Solution. *IEEE/ACM Trans. Netw.* **2013**, *21*, 1651–1665. [CrossRef]
30. Luo, J.; Su, X.; Liu, B. A Reinforcement Learning Approach for Multipath TCP Data Scheduling. In Proceedings of the 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 7–9 January 2019; pp. 276–280. [CrossRef]

31. Hurtig, P.; Grinnemo, K.J.; Brunstrom, A.; Ferlin, S.; Alay, Ö.; Kuhn, N. Low-latency Scheduling in MPTCP. *IEEE/ACM Trans. Netw.* **2018**, *27*, 302–315. [CrossRef]
32. Garcia-Saavedra, A.; Karzand, M.; Leith, D.J. Low Delay Random Linear Coding and Scheduling over Multiple Interfaces. *IEEE Trans. Mob. Comput.* **2017**, *16*, 3100–3114. [CrossRef]
33. Lim, Y.S.; Nahum, E.M.; Towsley, D.; Gibbens, R.J. ECF: An MPTCP Path Scheduler to Manage Heterogeneous Paths. In Proceedings of the 3th International Conference on emerging Networking Experiments and Technologies, Incheon, Republic of Korea, 12–15 December 2017; pp. 147–159.
34. Raiciu, C.; Barre, S.; Pluntke, C.; Greenhalgh, A.; Wischik, D.; Handley, M. Improving datacenter performance and robustness with multipath TCP. *ACM SIGCOMM Comput. Commun. Rev.* **2011**, *41*, 266–277. [CrossRef]
35. Walid, A.; Peng, Q.; Hwang, J.; Low, S. Balanced Linked Adaptation Congestion Control Algorithm for MPTCP. In *Internet Engineering Task Force, Internet-Draft Draft-Walid-Mptcp-Congestion-Control-04*; IETF: Wilmington, DE, USA, 2016.
36. Goyal, P.; Agarwal, A.; Netravali, R.; Alizadeh, M.; Balakrishnan, H. ABC: A Simple Explicit Congestion Control Protocol for Wireless Networks. *arXiv* **2019**, arXiv:1905.03429.
37. Nie, X.; Zhao, Y.; Li, Z.; Chen, G.; Sui, K.; Zhang, J.; Ye, Z.; Pei, D. Dynamic TCP Initial Windows and Congestion Control Schemes through Reinforcement Learning. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 1231–1247. [CrossRef]
38. Pokhrel, S.R.; Williamson, C. A Rent-Seeking Framework for Multipath TCP. *ACM SIGMETRICS Perform. Eval. Rev.* **2021**, *48*, 63–70. [CrossRef]
39. Liao, B.; Zhang, G.; Diao, Z.; Xie, G. Precise and Adaptable: Leveraging Deep Reinforcement Learning for GAP-based Multipath Scheduler. In Proceedings of the 2020 IFIP Networking Conference (Networking), Paris, France, 22–26 June 2020; pp. 154–162.
40. Silva, F.; Togou, M.A.; Muntean, G.M. AVIRA: Enhanced Multipath for Content-aware Adaptive Virtual Reality. In Proceedings of the 2020 International Wireless Communications and Mobile Computing (IWCMC), Limassol, Cyprus, 15–19 June 2020; pp. 917–922.