MPCC: Online Learning Multipath Transport

Tomer Gilad [1] Neta Rozen-Schiff[1] P. Brighten Godfrey[2] Costin Raiciu[3] Michael Schapira[1]
[1]Hebrew University of Jerusalem [2]UIUC [3]University Politehnica of Bucharest

ABSTRACT

Multipath transport, as embodied in MPTCP, is deployed to improve throughput and reliability in mobile and residential access networks, with additional use-cases including spreading load in data centers and WANs. However, MPTCP is fundamentally tied to TCP Reno's legacy AIMD algorithm, and significantly lags behind the performance of modern single-path designs. Consequently, MPTCP fails to achieve high performance in many real-world environments.

We present MPCC, a high-performance multipath congestion control architecture. To achieve our combined goals of fairness and high performance in challenging environments, MPCC employs online convex optimization (a.k.a. online learning). In experiments with a kernel implementation on emulated and live networks, MPCC significantly outperforms MPTCP.

CCS CONCEPTS

• **Networks → Transport protocols**.

KEYWORDS

Congestion Control, Multipath

ACM Reference Format:
Tomer Gilad [1] Neta Rozen-Schiff[1] P. Brighten Godfrey[2] Costin Raiciu[3] Michael Schapira[1]. 2020. MPCC: Online Learning Multipath Transport. In *The 16th International Conference on emerging Networking EXperiments and Technologies (CoNEXT '20), December 1–4, 2020, Barcelona, Spain.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3386367.3433030

1 INTRODUCTION

Multipath transport takes advantage of Internet path diversity by spreading traffic across multiple *subflows* for a single connection. Multipath TCP (MPTCP) [21], the principal embodiment of multipath transport, is the subject of numerous academic studies [11, 12, 16, 24, 29, 43–45, 47], and has also seen real-world adoption [31].

A central use case for multipath transport, and MPTCP specifically, is improving access throughput and reliability of devices (e.g., smartphones and tablets) using both WiFi and $4G/5G$ cellular interfaces by spreading traffic over the two network interfaces. Our experiments with MPTCP in emulated and live networks, however, reveal that in this scenario, which constitutes a primary motivation for MPTCP, and many others, MPTCP often exhibits bad performance. This is not surprising; the leading MPTCP variants [38, 51] are all extensions of a three-decades old congestion control scheme, namely, single-path TCP Reno, and consequently, inherit TCP's well-documented performance issues, including inability to adapt gracefully to changing network conditions, bad reaction to non-congestion loss, bufferbloat, RTT unfairness, and more [7, 13, 15].

Multipath congestion control is challenging. As in the single-path case, multipath congestion control must adjust sending rates in response to the prevailing network conditions. However, a multipath controller need also decide how to apportion its traffic among multiple paths to alleviate congestion. Furthermore, subflows belonging to the same transport-layer connection might compete with *each other* on different subsets of bottleneck links, rendering the congestion control challenge even more complex. Recent years have seen a surge of interest in novel approaches to high-performance congestion control [5, 10, 17, 18, 34], motivated by TCP's poor performance [17] and applications' ever-growing demands. Lately, researchers have also started investigating the applicability of these approaches to multipath congestion control [19, 26, 27, 42, 60, 61].

High-performance multipath transport design is more difficult than might first appear. First, simply running state-of-the-art single-path congestion control (e.g., PCC [17, 18], BBR [10], or Copa [5]) on each subflow fails to achieve fairness across competing single-path and multipath flows. Second, applying MPTCP's innovations to recent single-path protocols is nontrivial as MPTCP's design is closely tied to its congestion-window-based AIMD mechanism. We revisit MPTCP from an *online learning* (also termed online optimization [28]) perspective, which was recently applied to single-path congestion control [17, 18, 55]. This approach, termed Performance-oriented Congestion Control (PCC) in [17, 18], is compelling because it minimizes a priori assumptions about the network and has been shown to robustly provide high performance across many challenging environments [17, 18]. We present a multipath extension of PCC called *Multipath PCC (MPCC)*.

An MPCC connection repeatedly selects a *multidimensional* real-valued vector specifying its per-subflow rates. The implications for performance of sending at a combination of per-subflow rates are quantified by a *utility function*, which aggregates the contribution of the individual subflows to an overall performance score. The connection's choice of per-subflow rates, as a function of its past choices and derived utility values, is prescribed by an online learning algorithm with provable guarantees.

In tackling the challenge of meeting our goals for high-performance and fair multipath transport, we find that conventional approaches from online learning theory fail. To address this, we present a design that enables each of a connection's subflows to selfishly and asynchronously optimize a local, subflow-specific utility function. We prove that these local utility functions are sufficiently correlated across a connection's subflows to guarantee desired global outcomes.

In summary, our key contributions are:

- We leverage insights from online-learning to **design MPCC, a high-performance multipath transport**;
- **We present theoretical analyses of MPCC** establishing its convergence and fairness;
- **We implement MPCC in the Linux kernel** with sender-side changes only (the receiver-end remains legacy MPTCP), and we release our code as open source;
- **We evaluate MPCC through extensive simulations and experiments on a network emulator**. Our results show that MPCC can achieve high link utilization even with 25% the buffer size needed for MPTCP, MPCC achieves significantly higher utilization than MPTCP on links with random losses, can reduce self-induced latency by over 30%, and more closely tracks the optimum sending rate under variable network conditions;
- **We evaluate MPCC through live experiments on the Internet**. Our results show that MPCC provides an improvement (both in the mean and the median) of around 1.6× in terms of file download speed over MPTCP, with a speedup of up to 25× in some cases.

2 BACKGROUND AND GOALS

Single-path congestion control (e.g., TCP, BBR [10], PCC Vivace [17, 18], and Copa [5]) does not take advantage of multiple network interfaces and path redundancy. Multipath congestion control can achieve higher bandwidth by utilizing multiple paths, suffer lower loss and latency by shifting traffic to less congested paths, and be more resilient to failures by shifting traffic away from inactive paths.

Multipath congestion control must achieve three objectives [58]: (1) *improve throughput*; a multipath flow should provide at least as much throughput in aggregate as the best single-path transport on any of its paths, (2) *utilize the network efficiently* by shifting traffic away from congested paths, and (3) be no more aggressive than a single-path flow when several of its subflows share a bottleneck link.

Multipath TCP (MPTCP) [8, 58] extends TCP by introducing changes to connection setup and management to enable multiple *subflows* to utilize different network interfaces/paths. MPTCP also extends TCP to pace the sending rates of the individual subflows via per-subflow congestion windows [14].

Using a single-path congestion controller independently for each subflow would induce overly aggressive behavior when several of a connection's subflows traverse the same bottleneck, violating objective (3). Instead, MPTCP takes into account the rates of *all* subflows when changing the rate of a specific subflow. Specifically, as with single-path TCP Reno, an MPTCP subflow's window grows when a packet acknowledgement (ACK) is received, and is halved in the event of packet loss. The key difference is that instead of growing at a rate of one packet (MSS) per RTT, the pace at which each subflow's congestion window grows depends on the sum of all subflows' congestion windows [58]. This "coupling" between subflows ensures that MPTCP achieves its goals (see above) [22].

Four instantiations of MPTCP are implemented in the Linux kernel [13, 20, 25, 30, 36, 52, 58]. All are based on TCP and so share its many deficiencies. Indeed, evaluations of MPTCP variants [13–15] reveal rate-instability and unfairness, inefficient network utilization, and sometimes even worse performance than (single-path) TCP. Our aim is to satisfy the requirements from multipath congestion control within a design that addresses these issues.

3 MPCC OVERVIEW AND CHALLENGES

We next provide background on single-path online-learning transport and then present and discuss online-learning multipath transport.

3.1 Online-Learning Single-Path Transport

Online learning provides a powerful abstraction for reasoning about decision making under uncertainty. At time step $t = 1, 2, \ldots$, a decision maker selects a strategy s_t from a fixed set S. The implications of this choice are revealed after the fact, in the form of a utility value u_t. A rich body of literature in game theory and ML focuses on the design of algorithms that *provably* provide two types of guarantees: (1) *local* performance guarantees for the decision maker that hold even when the environment is *adversarial* ("regret minimization"), and (2) *global* desiderata such as quick convergence to an equilibrium when multiple decision makers interact [28, 62].

Performance-oriented Congestion Control (PCC) employs online learning [17, 18]. In PCC, time is divided into consecutive *monitor intervals* (MIs). In each MI t, the traffic sender (the decision maker) sends at a constant rate r_t (its strategy) and awaits feedback from the receiver in the form of selective acknowledgements [18] (SACKs). SACKs are used to compute statistics such as the goodput, loss rate, and latency, which, in turn, are aggregated into a utility value u_t reflecting the sender's local performance metric. The sender employs an online learning algorithm to adapt its rate in the direction, and to the extent, that is associated with higher performance.

The most recent manifestation of this approach, PCC Vivace [18], employs provably optimal (in the regret minimization sense) gradient-ascent-based optimization for this purpose. PCC protocols have been shown to significantly outperform TCP [17, 18]. Are similar performance gains over MPTCP achievable?

3.2 Online-Learning Multipath Transport

As in single-path PCC, in MPCC, the sender repeatedly selects sending rates, only now its choice of sending rates (strategy) is a *multidimensional* vector of real values $\vec{r} = (r_1, \ldots, r_d)$ such that each value r_i represents the sending rate of the i'th of its d subflows. We let $MPCC_d$ denote an MPCC connection with d subflows. The sending rates are constantly adjusted to optimize the sender's *utility*, which is derived from the performance of the different subflows.

Extending PCC's online learning scheme to the multipath context faces two nontrivial challenges, as discussed next.

Challenge I: What utility functions induce global stability, fairness, and efficiency? A utility function should capture the connection's performance objectives, but also guarantee convergence of competing MPCC connections to an efficient and fair outcome.

Challenge II: How should per-subflow rates be selected? Standard online-learning approaches fail. As we will show, applying standard approaches from online learning theory to adapt per-subflow sending rates yields undesirable effects, such as slowing down the frequency of rate selection at *all* subflows to that of the subflow with the *longest* RTT.

In our solution, each individual subflow optimizes a *subflow-specific* utility function. These are *loosely coupled*, in that subflows can adjust rates independently and asynchronously, yet are also sufficiently interrelated to guarantee that their (asynchronous) joint optimization induces desirable global outcomes. The decomposition of connection-level optimization into individual subflow-level optimizations draws inspiration from the literature on Network Utility Maximization (NUM) (see, e.g., [37, 48]).

4 FAILED TRY: CONNECTION-LEVEL RATE-CONTROL

Before delving into MPCC's design we first describe our first, failed attempt at online learning multipath transport. Examining this first design, which reflects a more obvious online learning approach, and the reasons for its inadequacy, is conducive for articulating our objectives and the challenges. In addition, our theoretical analysis of MPCC (in Section 5) builds on that presented below (in Section 4.1).

4.1 Connection-Level Utility Function

We define the following *connection-level* utility function, which consists of three components: a reward for throughput and penalties for loss and packet-delay.

Consider an multipath connection with d subflows (denoted by $[d] = \{1, \dots, d\}$) that sends at rate x_j on subflow $j \in [d]$. Let $X = (x_1, \dots, x_d)$ denote the vector of sending rates on all d subflows. Let the loss rate experienced by subflow j be denoted by $L_j \in [0, 1]$ and let the "latency gradient" [18] of subflow j, i.e., the slope of the increase in latency in the course of the monitor interval (MI) be denoted by $\frac{d(RTT_j)}{dT}$.

Intuitively, multipath transport should divert traffic from worse paths (paths with higher losses/delays) to better paths. This is achieved in our formulation by penalizing the connection according to the *worst* combination of loss and latency gradient across all of its subflows. The connection-level utility function U is defined as

$$U = (\Sigma_{j\in[d]} x_j)^{\alpha} - (\Sigma_{j\in[d]} x_j) \left(\max_{j\in[d]} (\beta L_j + \gamma \frac{d(RTT_j)}{dT}) \right), \quad (1)$$

where $0 \le \alpha < 1$, $\beta > 3$ and $\gamma \ge 0$. The above upper and lower bounds on α, β, and γ are needed to establish the correctness of our formal arguments below using our proof techniques. We leave the investigation of to what extent these can be relaxed to future research. Our theoretical results are proved for universal assignments of α, β, and γ.

This connection-level utility function provides a configurable way (through α, β, and γ) to balance different performance objectives.

Remark: When the number of subflows is 1 ($d = 1$), the connection-level utility function coincides with PCC Vivace's utility function [18].

4.2 Induced Network Utilization and Fairness

MPTCP theory does not mandate a specific notion of fairness. However, when several connections compete over the exact same set of links, any reasonable notion of fairness should stipulate that they should be allocated the same bandwidth. This is termed "resource pooling".

Parallel-(bottleneck)-link networks. When interaction between connections is more intricate, the desired outcome is no longer obvious. To guide our investigation of how network capacity is shared by multipath connections, we focus below on the class of parallel(-bottleneck)-link networks, which contains many classical topologies from MPTCP literature (see Section 7). We leave the analysis of other classes of networks for future research.

In a parallel-link network two vertices are connected by multiple (parallel) edges (links), and different multipath (and single path) connections use different subsets of these links. The parallel links model bottleneck links on which different subsets of connections compete (with other links on the connections' paths not being the bottleneck). We make no restrictions on the number of connections, the number of subflows per connection (which can vary across connections), the subset of links used by each connection, and link capacities.

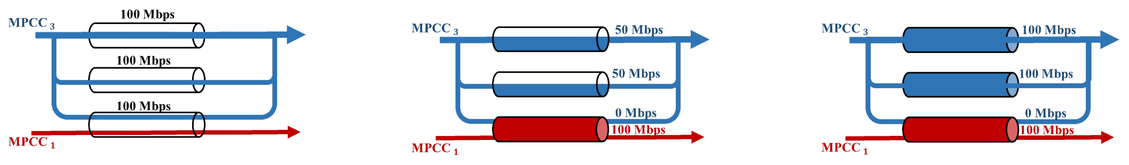
We show that, for this class of networks, our connection-level utility function induces equilibria that are fair in the classical (lexicographic) max-min fairness [6, 40, 46, 49, 50] sense.

Consider the example 3 parallel links network in Fig. 1a, in which a multipath connection with a single subflow ($MPCC_1$, which is equivalent to PCC Vivace) is competing with another multipath connection with 3 subflows ($MPCC_3$) on a single link, and the latter has also two links to itself. All links have capacity 100Mbps. Network capacity can be divided between the two connections in multiple ways (e.g., the 1-subflow MPCC sending at 50Mbps and the 3-subflow MPCC utilizing the rest of the network bandwidth).

Max-min fairness (MMF). MMF captures the desideratum that the bandwidth share assigned to the connection that is worst off be maximized. See [50] for a formal definition. Clearly, the 1-subflow MPCC connection can never be assigned more than the capacity of a single link (100Mbps). Thus, any global bandwidth allocation in which each of the MPCC senders sends at least at 100Mbps is MMF. This, however, includes the bandwidth allocation in Figure 1b, which is clearly suboptimal. To remedy this, the stronger notion of lexicographic max-min fairness (LMMF) [50] is considered.

Lexicographic max-min fairness (LMMF). Under LMMF, subject to maximizing the bandwidth assigned to the worst-off connection, the bandwidth assigned to the second-to-worst connection is also maximized, and so on. Let us revisit Fig. 1. In both the bandwidth allocation in Fig. 1b and the bandwidth allocation in Fig. 1c, the worst-off connection (the 1-subflow MPCC) sends at 100Mbps. Hence, both allocations are max-min fair. However, the bandwidth allocation of Fig. 1c also maximizes the assigned bandwidth of the second-to-worst connection (the 3-subflows MPCC, which is assigned 200Mbps, as opposed to 100 Mbps) and is hence LMMF. See [50] for a formal definition of LMMF.

The connection-level utility function induces LMMF equilibria. Consider homogeneous multipath connections interacting on a parallel-link network. We consider the equilibria in this setting, i.e., the global rate configurations such that no connection can improve its local (connection-level) utility by unilaterally changing its choice of per-subflow rates. We prove the following theorem.



(a) Three parallel-links topology **(b) Max-min fairness** **(c) Lexicograpic max-min fairness**

Figure 1: Fairness in a three parallel-links network topology

THEOREM 4.1. *In a parallel-link network, for any choice of number of multipath connections n, number of links m, link capacities, number of subflows d_i per each connection i, and assignment of subflows to links, any equilibrium is LMMF.*

See Appendix A for a sketch of the proof of Theorem 4.1.

4.3 Connection-Level Rate Selection

To select combinations of rates for its subflows, the connection can simply apply gradient ascent to its utility function, as follows. Time is divided into monitor intervals (MIs). In the beginning of each MI, the connection selects a combination of per subflow sending rates $(x_1, \ldots, x_d)$. The duration of an MI is set to be "sufficiently long" to collect statistics (throughput, latency gradient, loss rate) for each of the subflows. These statistics are then aggregated into a utility value as explained in Section 4.1. The choice of next rate for each subflow is guided by the (multi-dimensional) gradient of the utility function for the current rates, which informs the direction and extent to which the rate of each subflow should be adapted.

Such gradient ascent, for our choice of utility functions, induces the desirable local guarantees (regret minimization) and global guarantees (convergence to LMMF equilibria). However, our evaluation of this scheme revealed three significant deficiencies.

Obstacle I: Efficiently estimating the multidimensional gradient. Since different subflows of the same connection might share bottleneck links, probing rates for different subflows in parallel comes at the risk of very noisy estimations (as these potentially interfere with each other). Doing so sequentially, however, means spending too much time exploring rates. While theory-informed solutions are applicable, our experiments indicate that a good balance between good gradient estimation and fast reaction speed is elusive.

Obstacle II: Slow reaction. Requiring that an MI be sufficiently long to gather statistics for all of the subflows implies that MIs are at the timescale of the slowest RTT across all subflows.

Obstacle III: Wrong reaction. Since the connection-level utility penalizes loss and latency according the worst-case performance across all subflows, even if a subflow i is not experiencing high loss or delays, if another subflow j is experiencing excessive loss/delay, gradient ascent might prescribe that i not increase its sending rate (and even decrease it). While these effects are transient, in the sense that convergence to the "right" equilibrium is guaranteed, connections might exhibit erratic behavior during the convergence process.

5 MPCC DESIGN

MPCC's rate-control enables each subflow to independently and asynchronously optimize a local utility function, yet couples these subflow-specific optimizations in a manner that guarantees local and global desiderata. We next explain how this is accomplished

5.1 MPCC's Per-Subflow Utility Functions

Consider a specific connection i, and fix the sending rates of all subflows but subflow j, with c_{ik} denoting the sending rate of subflow $k \neq j$ (viewed as a constant). The utility function of subflow j, $U^{(j)}$, as a function of subflow j's sending rate x_{ij}, is as follows:

$$U_i^{(j)} = (\Sigma_{k \neq j \in [d_i]} c_{ik} + x_{ij})^{\alpha} - \beta(\Sigma_{k \neq j \in [d_i]} c_{ik} + x_{ij}) \cdot L_j$$
$$- \gamma(\Sigma_{k \neq j \in [d_i]} c_{ik} + x_{ij}) \cdot (\frac{d(RTT_j)}{dT}) \qquad (2)$$

where L_j and and $\frac{d(RTT_j)}{dT}$ denote the loss rate and latency gradient [18] experienced by subflow j, respectively, $0 \leq \alpha < 1$, $\beta > 3$ and $\gamma \geq 0$.

Subflow j's decisions depend only on its locally-perceived statistics. Optimizing the subflow-specific utility function does not involve learning the loss rates and latency gradients experienced by other subflows. The only information about other subflows required to compute subflow j's utility is their sending rates. Hence, short-RTT subflows need not wait for statistics resulting from the choices of rates of long-RTT subflows before adapting their rates, and so the MIs of different subflows need no longer be synchronized.

We prove that **decentralizing optimization across subflows in the above described manner still yields desirable global outcomes.**

THEOREM 5.1. *In a parallel-link network, for any choice of number of MPCC connections n, number of links m, link capacities, number of subflows d_i per each connection i, and assignment of subflows to links, any equilibrium induced by the MPCC per-subflow utility function in (2) is LMMF.*

A proof sketch for Theorem 5.1 appears in Appendix B.

5.2 MPCC's Per-Subflow Rate Control

In the beginning of each monitor interval, the subflow selects a sending rate for that MI. The MI is in the order of the RTT experienced by the subflow. When the MI is concluded and statistics about the performance of the subflow in that MI are gathered, the subflow's utility function is applied to compute the utility value corresponding to that MI. The subflow transitions between three states: (1) slow-start, (2) probing, and (3) moving.

Slow-start. In this phase, the subflow's sending rate is doubled in every MI until its utility decreases for the first time. Then, the subflow reverts to the previous sending rate and begins probing.

Probing. The goal of this state is to determine whether the sending rate should be increased or decreased, and to what extent. To this end, the gradient of the subflow-specific utility function at the current sending rate r is evaluated by testing a higher rate $r + \omega$ and a lower rate $r - \omega$. ω is not set to be a fraction of r, e.g., 1% of

r, but of the connection's total sending rate (sum of all subflows' sending rates). This is because the former can empirically result in inaccurate gradient estimations and, as a consequence, "getting stuck" at suboptimal global outcomes. Once the direction in which the rate should be adapted is decided, the moving phase starts.

Moving. In this phase, the subflow continuously adjusts its sending rate in the determined direction. The increase/decrease step size is determined based on the gradient of the utility function, which is inferred from the rates and corresponding utility values in the preceding two MIs. To speed up the subflow's rate adaptation, we incorporate mechanisms from PCC Vivace, namely, the rate amplifier, change bound (also expressed in terms of a fraction of the connection's total sending rate), and swing buffer. See [18] for details.

A subflow leaves the moving phase and re-enters the probing state when its utility function decreases for the first time.

Remark: rate-publication points. Computing the utility value of a subflow requires knowing the sending rates of the other subflows. To this end, each subflow "publishes", at the beginning of each MI, its chosen sending rate, making it available to the other subflows. The subflow treats the other subflows' published rates, as fed into a subflow's utility function, as constant throughout the gradient estimation (even if these were updated during its gradient estimation). This is to avoid the subflow rate-increase/decrease decisions being affected by changes in its utility that do not pertain to the subflow's own performance but to changes in the rates of others.

To illustrate MPCC's convergence, consider a variant of the topology in Figure 1 with only two parallel links, each of capacity 100Mbps. One MPCC connection, MPCC_2 uses both links, and is competing with a PCC (single-path MPCC) connection MPCC_1 on one of its links. As discussed in Section 4.2, in the LMMF outcome, each connection should fully utilize a different link. We plot the convergence process for the subflows on the shared link in Figure 2. The x axis is the rate of the subflow of MPCC_2, and the y axis is the rate of the PCC connection. The graph shows for each point (x, y) the gradient of the per-subflow utility functions on that link.

When the total sending rate on the shared link is below its capacity, both connections increase their rates according to their gradients. The MPCC connection derives less utility from additional bandwidth since it already has 100Mbps from its other subflow. Thus, PCC has a higher derivative and so increases its rate more rapidly. When the total sending rate on the shared link exceeds the link capacity, both connections reduce their sending rates (i.e., have negative derivatives). MPCC, which loses less in terms of utility when decreasing its rate, reduces its subflow's sending rate more quickly than PCC (see the diagonal downward arrows in Figure 2). In the equilibrium, marked by the red dot, PCC fully utilizes the link.

We prove that, as illustrated above, **MPCC is guaranteed to converge to LMMF equilibria on parallel-link networks.**

THEOREM 5.2. *In a parallel-links network, for any choice of number of MPCC connections n, number of links m, link capacities, number of subflows d_i per each connection i, and assignment of subflows to links, with utility function as in (2), MPCC converges to a LMMF global rate-configuration.*

See Appendix C for a sketch of the proof of Theorem 5.2.

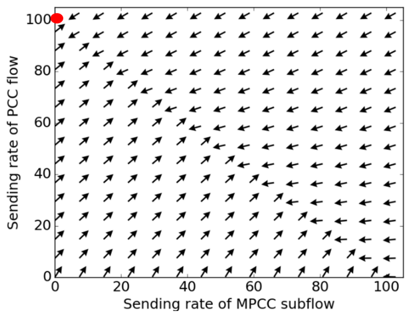


Figure 2: MPCC's and PCC's utility derivative convergence on a shared link

6 IMPLEMENTATION

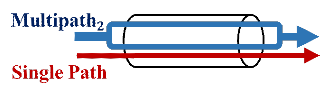
Our kernel implementation of MPCC involves two important changes to the MPTCP sender: (1) an MPCC congestion control kernel module, and (2) a scheduler for pacing-based multipath transport.

No changes to the MPTCP receiver. MPCC uses SACK feedback from standard MPTCP receivers.

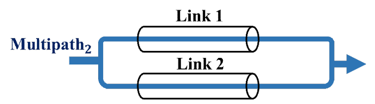
The MPTCP scheduler. The MPTCP scheduler is responsible for choosing which packets are transmitted on each of the connection's subflows. The scheduler is primarily intended for scenarios in which the connection is application-limited. The default MPTCP scheduler [1] sticks with the subflow with the lowest RTT until its congestion window is exceeded. When a connection continuously sends traffic (that is, is not application-limited), the subflows become ACK-clocked; once an ACK is received, a new packet is sent.

The MPTCP scheduler is incompatible with rate-based (pacing-based) congestion control. While the MPTCP scheduler is suitable for window-based protocols, we find that it is ill suited for rate-based protocols (like PCC and BBR), in which the congestion controller sets explicit rates. In PCC Vivace [18], for instance, the window is deliberately set to be high and is not expected to be exceeded unless something unexpected occurs. This implies that if (single path) PCC Vivace is employed for adapting rates for each of the subflows, all packets will be assigned by the default scheduler to the lowest-RTT subflow (since the condition pertaining to the window will never be met). This can, of course, result in other subflows sending at low rates, or not at all, and so in low utilization of network capacity.

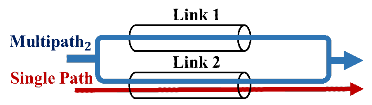
A scheduler for rate-based multipath congestion control. To accommodate rate-based multipath congestion control, we changed the default MPTCP scheduler as follows: a subflow is marked as unavailable for sending when the subflow has at least 10% of the packets required to maintain the current sending rate for the duration of an RTT already queued for sending. The choice of 10% was guided by an empirical investigation. When the threshold is set to be high, a subflow will be deemed available even if many packets are queued for sending on that subflow. This will lead MPCC to assign too much weight to low-RTT subflows, wasting available capacity on other subflows. If, in contrast, the threshold is set to be too low, subflows will become unavailable even when very few packets are scheduled for sending on these subflows, leading to packets being sprayed on subflows uniformly, which is undesirable for short-lived flows. Our experiments with pacing-based protocols suggests that



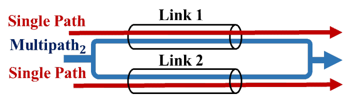
(a) Multipath connection *vs.* single path connection ("single link MP-SP").



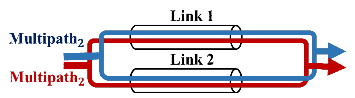
(b) One Multipath connection.



(c) Multipath connection *vs.* single-path connection ("two links MP-SP").



(d) Multipath connection *vs.* two single-path connections ("two links MP-SP-SP").



(e) Two Multipath connections.

Figure 3: Evaluated 1- and 2-link networks. Default link latencies, bandwidths, and buffer sizes are $30ms$, $100Mbps$ and $375KB$ (the BDP), respectively

setting the threshold to be 10% strikes a reasonable balance between these two extremes. We ran experiments on the Emulab network emulator [56] to verify that the above changes to the scheduler indeed yield high bandwidth utilization. In experiments with a single multipath connection sending traffic across two parallel 100 Mbps links using single-path BBR to adjust rate for each subflow, the goodput increased from 148.2 Mbps (with the default scheduler) to 179.4 Mbps.

Open-sourced MPCC and scheduler kernel modules. To evaluate MPCC, we have implemented it and our scheduler as MPTCP Linux kernel modules. Our kernel implementation extends the PCC Vivace kernel module [33] to incorporate the capabilities discussed in Section 5.2, including specifying per-subflow utility functions, realizing rate-publication points, and implementing probing and change bounds that are a fraction of the sum of all subflows' sending rates. The code for our kernel modules is available at [2].

7 EVALUATION

7.1 Evaluation Framework

MPTCP variants. We compare MPCC to the MPTCP variants implemented in the Linux kernel— Lia [51], Olia [38, 39], Balia [48, 54], and wVegas [9]—and to using *single path* congestion control (namely, Reno [20], Cubic [23], and BBR [10]) for each subflow.

Two MPCC utility functions. We evaluate MPCC for two different choices of utility functions: one with the penalty for latency increase set to 0, which we term MPCC-loss, and one with non-zero penalty, which we term MPCC-latency. Specifically the parameters in the utility function for both MPCC-loss and MPCC-latency are $\alpha = 0.9$ and $\beta = 11.35$. γ is set to be 0 and 1 for MPCC-loss and MPCC-latency, respectively. These specific parameters are chosen so that MPCC$_1$ match the specification of PCC Vivace evaluated in [18].

Emulations and live experiments. We used Emulab [56] to emulate different networks. We also ran live experiments on the Internet by downloading files from different locations in the cloud to devices with Wifi and cellular interfaces in residential networks. We use MPTCP kernel version 0.95 in our experiments for all of the sending/receiving nodes. Our Emulab experiments used Ubuntu 16.04 running on d710/d430 machines. We used bridge nodes with ipfw to control the link properties.

Schedulers. In our experiments, our scheduler (in Section 6) is used for rate-based schemes (including MPCC) and the default scheduler is used for window-based schemes (all MPTCP variants).

OS settings. To avoid flow control issues, we set the buffers at the sender and receiver to be 300MB. We also disable the MPTCP checksum calculation in our experimental setup to reduce the per-packet computational overhead.

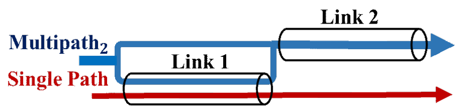
7.2 Emulab Experiments

Considered network topologies: parallel and non-parallel-link networks. Fig. 3 presents 5 parallel-link networks emulated in our experiments. Fig. 4 presents two considered non-parallel-link networks from prior studies of multipath congstion control, namely, the "OLIA Topology" [38] and "LIA Topology" [58]. Unless stated otherwise all link latencies, bandwidths, and buffer sizes are $30ms$, $100Mbps$ and $375KB$ (the Bandwidth-Delay Product), respectively.

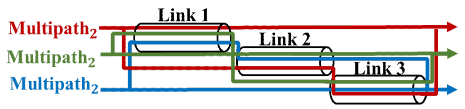
7.2.1 **Better performance for shallow buffers.** Fig. 5a plots the average goodput of the single multipath connection in Fig. 3b in 5, 200 second long iperf3 [3] runs, with the first 30 seconds omitted to allow for convergence to steady-state. The buffer size of link 2 is set to be the BDP, i.e., 375KB, and the buffer size of link 1 is varied within the range $[3, 375]$ KB. MPCC achieves high utilization even with buffer size as low as 9KB. Lia, Olia and Balia require about 60KB to reach the same level of network utilization. wVegas fails to utilize the link well across the entire range of buffer sizes.

We repeat the above experiment, only now the multipath sender competes with a single path sender (i.e., PCC Vivace for MPCC and TCP Reno for MPTCP) on link 2, as described in Fig. 3c. We hypothesize that when a multipath protocol fails to utilize link 1 well, this will result in it being more aggressive on link 2 and, consequently, the single path connection achieving lower goodput. Fig. 5b plots the throughput of the single path connection (on link 2). For all MPTCP variants, with the exception of wVegas (which achieves very low throughput in general), the goodput of the single path TCP connection drops when the size of the buffer on the *other* link (link 1) gets too small. Observe also that even when the average network utilization of an MPTCP variant, as plotted in Fig. 3b, matches that of MPCC, MPTCP is still less fair.

7.2.2 **Higher resilience to non-congestion loss.** Congestion control protocols sometimes experience loss that is not derived from

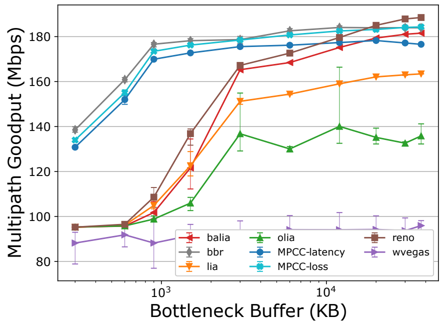


(a) The "OLIA Topology" from [38].

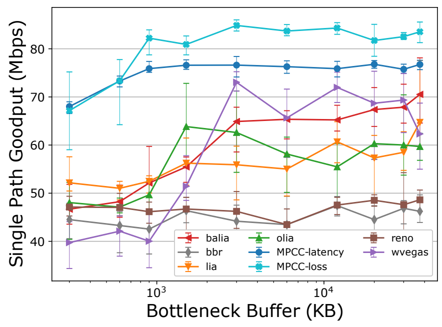


(b) Three links topology with three Multipath connections (the "LIA Topology" from [58]).

Figure 4: Evaluated non-parallel topologies. Default link latencies, bandwidths, and buffer sizes are $30ms$, $100Mbps$ and $375KB$ (the BDP), respectively.

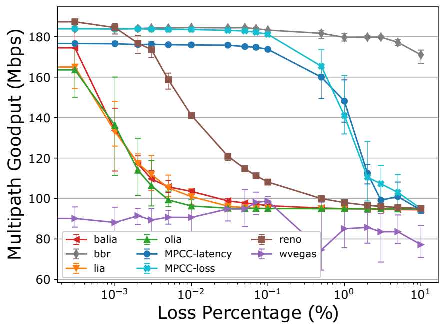


(a) Multipath connection's throughput in network 3b

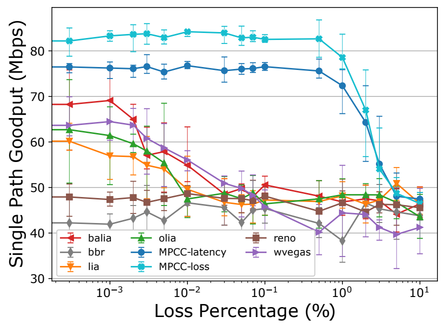


(b) Single path connection's throughput in network 3c

Figure 5: Network utilization with shallow buffers (log scale). Error bars show the minimum and maximum recorded over 5 experiments.



(a) Multipath connection's throughput in network 3b



(b) Single-path connection's throughput in network 3c

Figure 6: Throughput of multipath and single path senders in the presence of random loss (log scale). Error bars show the minimum and maximum recorded over 5 experiments.

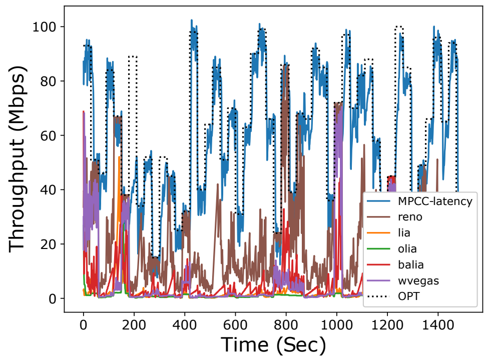


Figure 7: The throughput of a multipath connection's subflow on link 1 in the network in Fig. 3c under changing network conditions. The dashed black line is the optimal throughput.

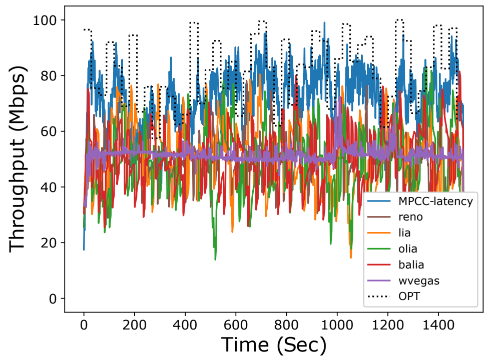


Figure 8: The throughput of a single path connection in the network in Fig. 3c. The dashed black line represents its fair (in the LMMF sense) share over time.

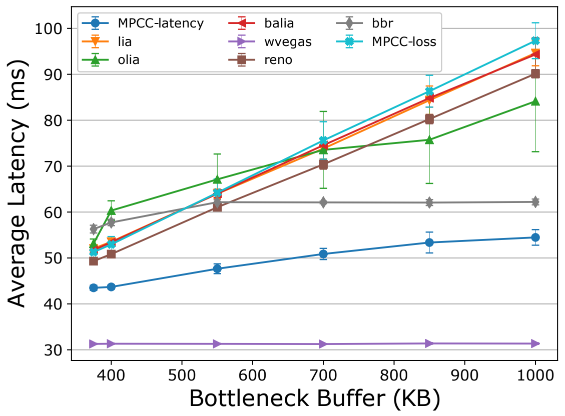


Figure 9: MPCC maintains lower self-induced latency as the bottleneck link's buffer size increases. Error bars indicate the standard deviation of the average latency.

congestion (e.g., due to handover between mobile base stations, physical layer corruption, routing changes, and more). To capture the resilience of different multipath protocols to non-congestion loss, we present the results of an experiment similar to that described in Section 7.2.1, only now instead of varying the buffer size, we introduce random loss on link 1. As shown in Fig. 6a, Lia, Olia, and Balia, which react to packet loss by multiplicatively decreasing the congestion window (similarly to TCP Reno) underutilize the network even for loss rate as low as 0.001%, dropping below 80% utilization. wVegas fails to achieve high throughput across the entire range of random loss rates. MPCC, in contrast, maintains higher throughput across the entire range, and drops below 80% utilization only upon experiencing loss rate 500 times higher than that which has the same effect on MPTCP. As expected, the multipath connections employing a single path congestion controller, namely, Reno or BBR, for each subflow are also quite resilient to random loss; BBR is known to be highly resilient even to high random loss [10] whereas single path Reno, while halving its congestion window upon loss, increases the window more quickly than its multipath extensions (see Section 2). Similarly to our results in Section 7.2.1, MPCC's consistently higher utilization on link 1 translates to being more fair to a single path connection on link 2, as shown in Fig. 6b.

7.2.3 **Better adaptation to network changes.** Fig. 7 presents the throughput of a multipath subflow on link 1 in the network in Fig. 3c. The link's bandwidth, latency, and loss rate are assigned random values every 30 seconds from the ranges $10-100$ Mbps, $10-100$ ms, and $0.01\% - 0.1\%$, respectively. The buffer size is 375KB. The dashed black line in the figure represents the link bandwidth at each point in time. As seen in Fig. 7, MPCC's throughput approximates this optimum better than all the other multipath protocols. Fig. 8 shows the implications for the single path connection's throughput. Observe that the single path PCC connection's throughput most closely approximates its fair share (in the LMMF sense) on the link.

7.2.4 **Lower latency.** Lia, Olia and Balia are loss-based. In contrast, MPCC-latency optimizes per-subflow utility functions that penalize increases in latency (Section 4.1). To demonstrate the implications for performance, we evaluate MPCC and MPTCP on the network in Fig. 3e. We use the ss utility [4] to sample the smoothed RTT of each connection every 0.1 seconds for 200 seconds and present in Fig. 9 the average latency over 5 runs of iperf3 for each sender as the bottleneck buffer size increases. Our focus is on buffers whose sizes exceed the BDP as our goal is to quantify the self-induced latency when the network is fully utilized.

MPCC-latency achieves better RTTs than Lia, Olia and Balia for all evaluated buffer sizes. We point out that this lower latency comes at the cost of slightly lower link utilization (as will be discussed in Section 7.2.5), since MPCC-latency avoids filling buffers. While wVegas yields low RTTs, its utilization is extremely low, as will be discussed in Section 7.2.5. Since BBR also incorporates latency measurements into its decision making, the multipath connection running BBR for each of its subflows fares better than the loss-based protocols, though still worse than MPCC-latency.

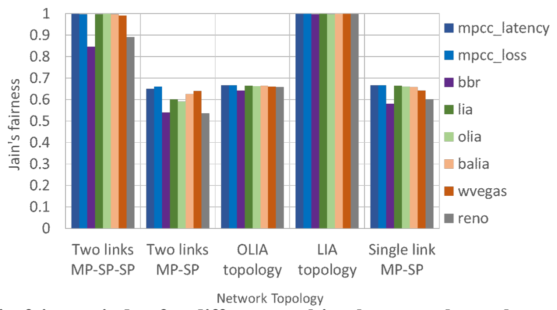
7.2.5 **Comparable convergence.** As shown in Section 7.2.1, when buffers are small, MPTCP suffers from low link utilization. In our experiments, on links with shallow buffers, MPTCP converges to bad stable states (low utilization, not fair), and sometimes does not even converge. Thus, to compare MPCC and MPTCP's convergence we focus below on the scenario that buffers are sufficiently deep to support MPTCP convergence. Specifically, in all of the examples below the buffer sizes on all links match the BDP (375KB).

We evaluate convergence on 5 of the networks in Fig. 3 and Fig. 4b, using 10 iperf3 200-seconds long runs, again omitting the first 30 seconds. Fig. 10b plots the average ratio between the total goodput across all connections and the total network capacity, and Fig. 10a shows how fairly network capacity is distributed using Jain's fairness index [32]. The results show that even in ideal conditions for MPTCP, MPCC provides comparable utilization and fairness. Multipath connections that use single path (BBR/Reno) for each connection are naturally the least fair.

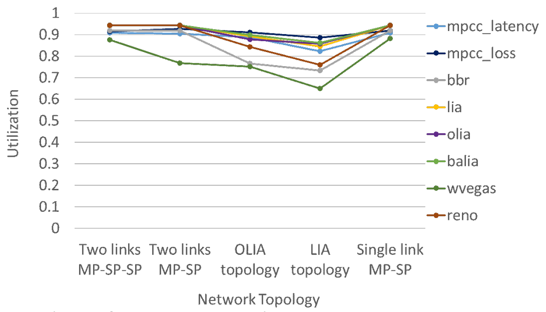
In our experiments, MPCC provides comparable convergence rates to MPTCP, but with lower rate-jitter. See Fig. 11, which depicts the convergence of MPCC-latency and Balia on topology 3c.

Observation: Sometimes employing a single-path congestion controller for each subflow is bad for performance (not only fairness). As shown in Fig. 10b, running Reno or BBR independently for each subflow provides worse overall goodput than MPCC and even MPTCP on the OLIA and LIA topologies. To gain intuition for this phenomenon (already observed in [14, 58]), consider the OLIA Topology. The gootput-maximizing (and fair) outcome is the single path connection fully utilizing Link 1 and the multipath connection fully-utilizing Link 2. In this outcome the total goodput is 200Mbps. However, if the two connections equally split Link 1's bandwidth (getting 50Mbps each) and the multipath connection takes over the remaining bandwidth of Link 2, the overall goodput is 150Mbps. Using single-path congestion control independently per subflow will approximate the second outcome.

7.2.6 **TCP friendliness.** To investigate how the multipath protocols interact with legacy TCP, we repeat the experiments in Sections 7.2.1 (for varying buffer sizes) and 7.2.2 (for varying random loss rates), with the exception that the single-path sender is now running TCP Cubic. Recall that in these experiments, a multipath connection sending on two links is competing with a single-path connection on one of these links. Since the purely loss-based PCC

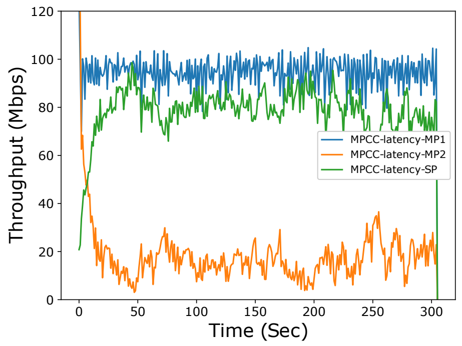


(a) Jain fairness index for different multipath protocols on the network topologies in Fig. 3,4

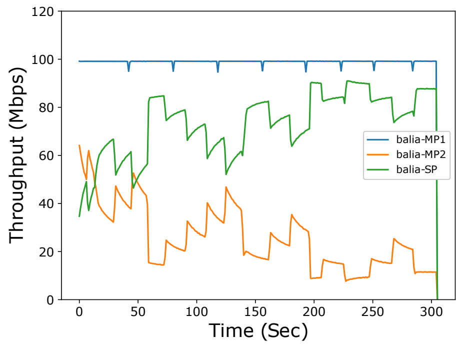


(b) Comparison of the average ratio between the total goodput and the total network capacity on the network topologies from Fig. 3,4

Figure 10: Fairness (10a) and normalized total goodput (10b) for different multipath protocols on topologies from Fig. 3 and Fig.4b.

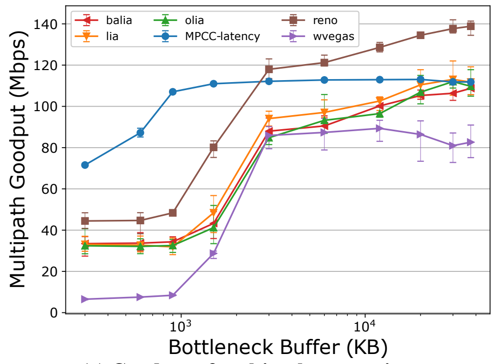


(a) MPCC-latency convergence on Topology 3c

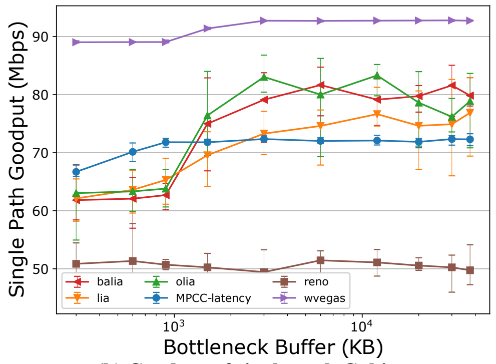


(b) Balia convergence on Topology 3c

Figure 11: Example of MPCC's and Balia's convergence on network 3c. The multipath's subflow 2 (orange) is on the same link as the single path flow (green).



(a) Goodput of multipath connection.



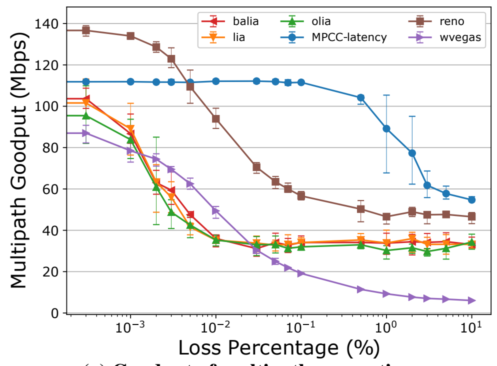
(b) Goodput of single-path Cubic.

Figure 12: Goodput of Multipath sender and single-path TCP Cubic in topology 3c when varying buffer size of link 1

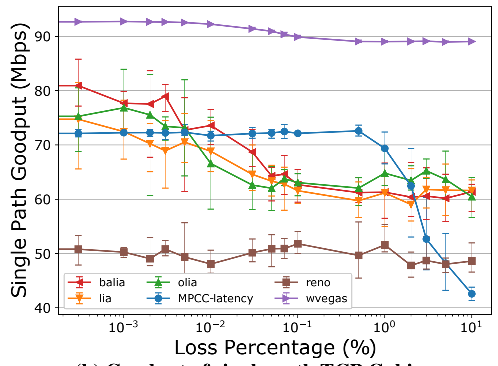
Vivace (which is equivalent to the purely loss-based MPCC$_1$) is not friendly towards legacy TCP [18], we focus on the latency-sensitive variant of MPCC. We aim to investigate to what extent this variant of MPCC, shown to provide good performance in both emulated networks (see Section 7.2.7) and live networks (see Section 7.3) harms legacy TCP. We leave the question of how MPCC loss can be altered to achieve TCP friendliness (without suffering too costly a price in terms of performance) to future research. Figs. 12 and 13 show the goodput of the multipath sender and the TCP Cubic sender. Observe that in all experiments, when competing with MPCC or any variant of MPTCP, the single-path TCP Cubic attains well over 50% of the link's bandwidth (typically over 70%). Thus, in these scenarios, competing against a multipath connection (including MPCC) is better for single-path TCP Cubic, performance-wise, than competing against a single-path TCP Reno.

Bandwidth (Mbps)	50	500	
Latency (ms)	10	100	
Loss rate	0%	0.1%	0.001%
Buffer (KB)	50	700	

Table 1: Possible link parameters

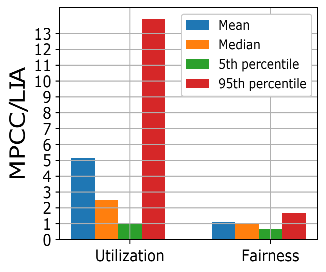


(a) Goodput of multipath connection.

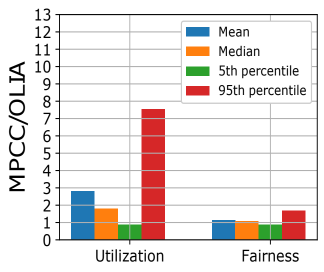


(b) Goodput of single-path TCP Cubic.

Figure 13: Goodput of multipath sender and single-path TCP Cubic in topology 3c for varying loss rate on link 1

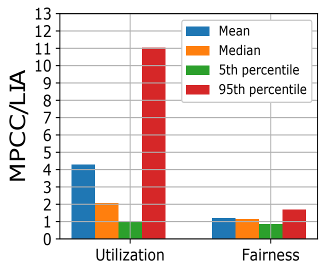


(a) MPCC compared to LIA

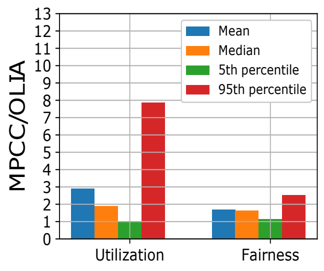


(b) MPCC compared to OLIA

Figure 14: Comparing MPCC to LIA and OLIA for varying link parameters on topology 3c.



(a) MPCC compared to LIA



(b) MPCC compared to OLIA

Figure 15: Comparing MPCC to LIA and OLIA for varying link parameters on topology 3d.

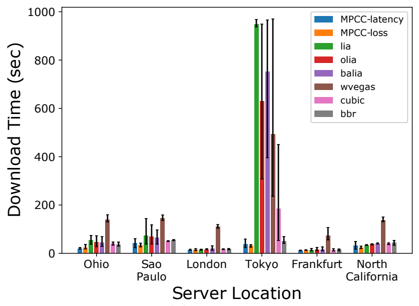
7.2.7 **Varying the network configuration.** So far, our experiments used specific, default assignments to network parameters (link latencies, bandwidths, and buffer sizes of $30ms$, $100Mbps$ and $375KB$, respectively) as the base network configuration. To gain further insights into the impact of changes to these parameters, we consider the two-link MP-SP network (Fig. 3c) and compare MPCC to LIA and OLIA for all choices of network parameters in Table 1. This amounts to $24^2 = 576$ different network configurations (24 different options for each of the two links). In Fig. 14(a) we present the mean, median, 5th percentile and 95th percentile ratio between the average bandwidth utilization of MPCC-latency and LIA across all 576 experiments. Fig. 14(a) also presents the ratio between MPCC-latency and LIA in terms of the Jain fairness index. Fig. 14(b) plots the utilization and fairness ratios for MPCC-latency and OLIA. All results are averaged over 5 runs per network configuration.

As can be seen in the figure, MPCC achieves significantly better utilization than both LIA and OLIA in terms of the mean, median, and the 95th percentile. MPCC also exhibits somewhat better fairness in the mean, median, and the 95th percentile. To characterize the circumstances in which MPCC is better/worse than the two considered MPTCP variants, we examine the network configurations for which the best/worst ratios are obtained. Not surprisingly, the best ratios for MPCC, in terms of both utilization and fairness, are obtained for high non-congestion loss on link 1 (the non-shared link). This is consistent with our results in Section 7.2.2 above. Interestingly, however, the worst ratios for MPCC occur when the bandwidths of the two links are non-identical. When the bandwidth of the non-shared link (Link 1) is 500Mbps and the bandwidth of the shared link is 50Mbps, the increases to MPCC's rate on the lower-bandwidth link, which are set to be fraction of the *total* sending rate across all its subflows (see Section 5.2), are too big, leading MPCC to often overshoot that link's bandwidth. This results in packet losses and delays that, in turn, slow down both senders on the shared link. Moreoever, the multipath sender ends up using less bandwidth than available on the *non-shared* link because the losses on *the other* link, which involve many retransmits, prevent the receiver from accepting more packets before previously lost packets are successfully re-sent.

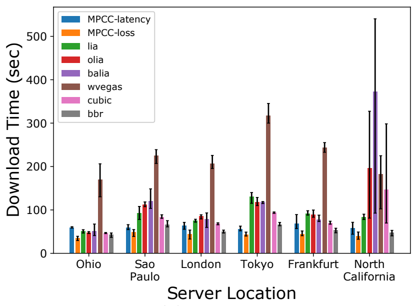
Repeating the same set of experiments for network environment 3d yields similar results, as shown in Fig. 15. Since the multipath connection now competes with a single-path connection on each of the links, the imbalance between its achieved throughputs on the two links is less severe than in the previous set of experiments (where it had one link all to itself). This accounts for the improved utilization and fairness ratios in the 5th percentiles.

7.3 Experiments for AWS-to-Residential

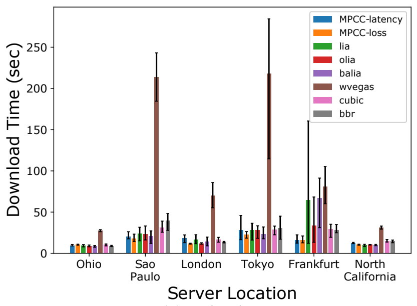
A common use case for MPTCP is devices with WiFi and cellular interfaces. We test the time it takes to download a 75MB file from cloud-based servers in various locations around the world to computers in three home network in Israel, Boston and Illinois with WiFi access and an additional cellular interface. In these live experiments, a mobile phone provided cellular connectivity to a PC using USB tethering, and the PC was connected to a residential WiFi access point on another interface. The PC was configured to create one subflow on each interface. The average of 3 runs, for each server-home pair with the different multipath protocols run in



(a) Israel.

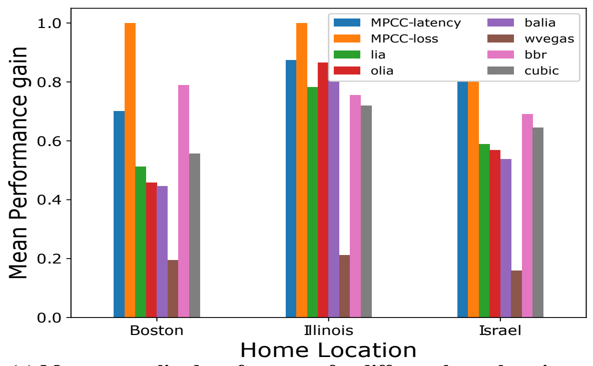


(b) Boston.

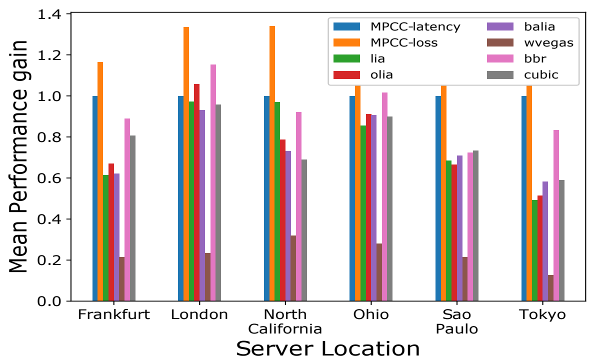


(c) Illinois.

Figure 16: Download times of a 75MB file from different AWS servers to residential networks in different locations. Error bars indicate minimum and maximum times measured.



(a) Mean normalized performance for different home locations.



(b) Mean normalized performance for different servers.

Figure 17: Mean performance normalized to MPCC-latency. Higher is better.

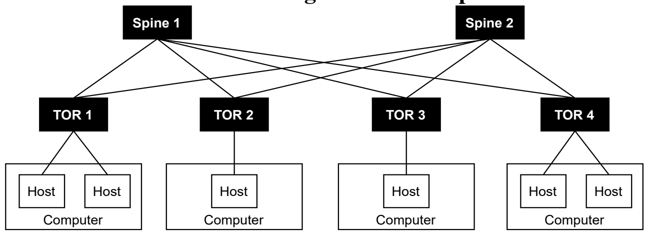


Figure 18: Our data center testbed topology.

round-robin order, appears in Fig. 16. Fig. 17 shows an aggregated and normalized breakdown of these results where each bar is the ratio between MPCC's mean download time and another protocol's mean download time, and so higher is thus better.

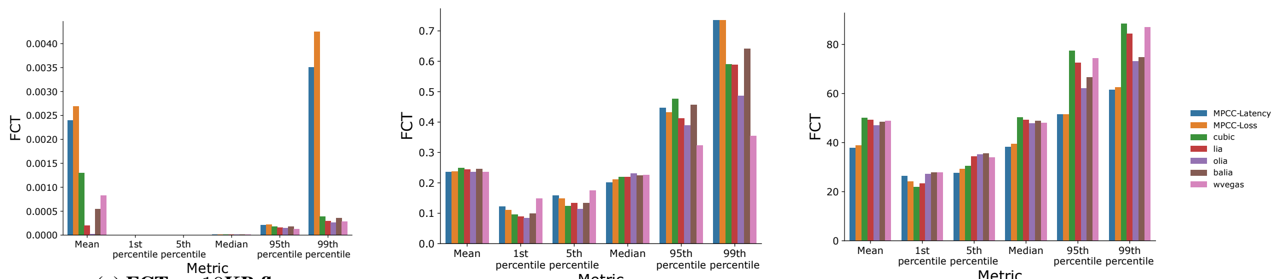
Observe that (1) MPCC-loss consistently matches or outperforms the other protocols, achieving download times up to 25X faster than Lia, Olia and Balia. When averaging the performance gains across all experiments MPCC-loss is 1.59X faster than Lia, 1.58X faster than Olia and 1.62X faster than Balia, and in the median MPCC-latency is 1.18X faster than lia, and 1.29X faster than Olia and Balia; (2) While MPCC-latency sacrifices some throughput for improved latency, it still does not lag far behind MPCC-loss; (3) MPCC's performance gains over MPTCP increase as the distance (and so the BDP) grows; (4) Employing single-path BBR for both subflows is frequently better than the MPTCP variants not only because single-path BBR typically outperforms single-path TCP [10], but also because this results in increased aggression towards competing connections. MPCC, while also designed for fairness, still achieves high performance, and is frequently better than BBR.

While some of MPCC-loss' performance gains can be attributed to its increased aggression towards persistent legacy TCP connections (see discussion in Section 7.2.6), the performance improvements achieved by MPCC-latency (which exhibits TCP friendliness in our experiments) are derived from MPCC's online learning algorithmic framework.

7.4 Experiments on a Small Data Center Testbed

Our MPCC design primarily targets scenarios like utilizing multiple network interfaces in last-mile network environments. We do, however, take a first step towards analyzing MPCC in another common use-case for MPTCP: data centers. Data center networks are very different from those considered thus far, with significantly higher bandwidths, significantly lower end-to-end latency, and support for explicit network feedback. We conduct experiments with our implementations of MPCC-loss and MPCC-latency on a small data center network testbed. We leave the adaptation of MPCC to data centers and extensive evaluation of this use case to the future.

Our testbed consists of 6 switches and 6 hosts. Switches comprise a 2-layer Clos network, as in Fig. 18, with 25Gbps DAC cables. Hosts run on 4 dual-homed Xeon Silver processors with 128GB RAM each. Shortest-paths are hardcoded into the switches with ECMP hashing used to choose between them. We ran the following experiment, which evaluates different multipath protocols under heavy and dynamically changing load: first, 15 10GB flows and 35 10MB flows were *simultaneously* initiated from each host to other, randomly chosen hosts in the network. Then, at each second of the first minute of the experiment, each host started a single 10KB flow



(a) FCT on 10KB flows. **(b) FCT on 10MB flows** **(c) FCT on 10GB flows**

Figure 19: FCT (sec) measurements on our datacenter testbed.

to a randomly chosen host. We repeated this experiment 10 times. All flows were sent via multipath connections, each with 3 subflows.

Fig 19 plots our results for flow completion time (FCT). Observe that MPCC achieves worse FCTs for the short flows, comparable FCTs for the medium flows, with slightly worse performance in the tail, and better FCTs for the long flows. Thus, the benefits of MPCC in this specific data center environment apply only to the long flows (10GB). We conjecture that addressing this involves tackling two issues: (1) MPCC not sufficiently ramping up its rate at the early stages of a connection, and (2) MPCC inducing packet retransmissions in the presence of heavy and dynamic traffic, which primarily slows down the shorter connections.

8 RELATED WORK

In recent years, several new paradigms for overcoming TCP's limitations have been proposed. Remy [57] and PCC [17, 18] reflect offline and online optimization approaches, respectively. BBR [10], models the network pipe as a single (bottleneck) link and tracks that link's bandwidth and RTT. Copa [5] optimizes rate selection within a model in which packet arrival at the connection's bottleneck link is a Poisson distribution.

MPTCP variants include Linked-Increases Algorithm (LIA) [51], Opportunistic Linked-Increases Algorithm (OLIA) [38], Balanced Linked Adaptation (Balia) [54], MPCUBIC [41], and wVegas. LIA, OLIA and Balia adapt TCP Reno to the multipath context. MPCUBIC and wVegas [9] extend TCP Cubic and TCP Vegas, respectively. See [59] for a survey and [13–15] for evaluation results.

Network Utility Maximization (NUM) [35] is a prominent framework for TCP protocol design, which has also been applied to the multipath context [25, 36]. In NUM, an end-to-end congestion control protocol and queuing policies at routers are derived from the primal-dual solution to a centralized convex optimization problem. Our decomposition of connection-level optimization into individual per-subflow optimizations is inspired by ideas from NUM literature (see, e.g., [37, 48]). While NUM-induced schemes provably converge to a global optimum in arbitrary network topologies, this involves strong assumptions that are often violated in real-world environments such as last-mile networks (MPTCP's main use-case), e.g., (1) *all* connections run the prescribed protocol, (2) *all* routers in the network employ the prescribed queueing policy, (3) *all* connections are sufficiently long-lived for a steady state to be approximated, and more. Online-learning-based congestion control, in contrast, provides meaningful local performance guarantees for the individual connection ("no regret" [18]) even when treating the network as a black box, while also reaching desired global equilibria *if* all connections are long-lived and employ the protocol. See [53] for a discussion in the context of single-path congestion control.

Recently, multipath congestion control has been revisited from two new perspectives: (1) adapting BBR to this context [26, 27, 61], and (2) applying reinforcement learning [19, 42, 60]. We leave the thorough comparison of our online learning approach and these approaches to future research. We point out, however, that, as discussed above, a key strength of our approach are its *provable* local and global guarantees (no regret and global convergence). In addition, unlike our approach, the proposed reinforcement learning approaches to multipath congestion control rely on offline training of the multipath congestion controller on empirical data, and so involve the standard costs associated with such approaches, including bad performance when the training environment and operational environment differ and the high sample complexity of deep reinforcement learning (need for many data points for effective learning).

9 CONCLUSION AND FUTURE RESEARCH

We revisited multipath congestion control and advocated its reexamination through the lens of online learning. We presented MPCC, a promising first step in this direction. Our evaluation of a kernel implementation of MPCC suggests that MPCC constitutes a promising alternative to MPTCP.

Our investigation of MPCC motivates further exploration of its theoretical and empirical guarantees and of how our design can be further improved. Specifically, our theoretical results leave open the question of obtaining provable guarantees that extend beyond the domain of parallel-link networks. In addition, limitations of our MPCC implementation that deserve careful attention include suboptimal performance on network paths that greatly differ in terms of available bandwidth (see Section 7.2.7) and suboptimal flow-completion-times for short flows (see Section 7.4). Lastly, our evaluation (with the exception of the experiments on the data center testbed), focused on bulk transfers. Additional measurements of MPCC's performance under other traffic conditions is required to evaluate how MPCC affects different kinds of applications.

ACKNOWEDGEMENTS

We thank the anonymous shepherds and reviewers for many valuable comments. Brighten Godfrey and Michael Schapira were partly supported by a an NSF-BSF CNS Award (NSF 2008971, BSF 2019798). Costin Raiciu was partly supported by research gift funding from Huawei. Michael Schapira was partly supported by the ISF.

REFERENCES

[1] 2017. MultiPath TCP - Linux Kernel implementation. https://multipath-tcp.org/pmwiki.php/Users/ConfigureMPTCP. (2017).

[2] 2019. MPCC prototype open source release. https://github.com/mpccopensource/mpcc_release. (2019).

[3] 2019. MultiPath TCP - Linux Kernel implementation. = https://github.com/esnet/iperf. (2019).

[4] 2019. ss(8) - another utility to investigate sockets. = http://man7.org/linux/man-pages/man8/ss.8.html. (2019).

[5] Venkat Arun and Hari Balakrishnan. 2018. Copa: Practical Delay-Based Congestion Control for the Internet. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. Renton, WA, 329–342.

[6] Dimitri Bertsekas and Robert Gallager. 1987. *Data Networks.* Prentice-Hall, Inc.

[7] P. A. Bhat and G. Talmale. 2014. MPTCP combining congestion window adaptation and packet scheduling for multi-homed device. In *International Conference for Convergence for Technology-2014*. 1–6.

[8] Olivier Bonaventure, Mark Handley, and Costin Raiciu. 2012. An overview of Multipath TCP. *USENIX login;* (October 2012).

[9] Yu Cao, Mingwei Xu, and Xiaoming Fu. 2012. Delay-based congestion control for multipath TCP. In *2012 20th IEEE International Conference on Network Protocols (ICNP)*. 1–10.

[10] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. 2016. BBR: Congestion-Based Congestion Control. *Queue* 14, 5, Article 50 (Oct. 2016), 50:20–50:53 pages.

[11] Yung-Chih Chen, Yeon-sup Lim, Richard J. Gibbens, Erich M. Nahum, Ramin Khalili, and Don Towsley. 2013. A Measurement-based Study of MultiPath TCP Performance over Wireless Networks. In *Proceedings of the 2013 Conference on IMC*. New York, NY, USA, 455–468.

[12] Yung-Chih Chen, Don Towsley, and Ramin Khalili. 2014. MSPlayer: Multi-Source and multi-Path LeverAged YoutubER. In *Proceedings of the 10th ACM CoNEXT '14*. 263–270.

[13] A. Chodorek and R. R. Chodorek. 2017. Analysis of a high-performance mobile access to a computing cloud using MPTCP protocol and IEEE 802.11ac network. In *2017 15th International Conference on ITS Telecommunications (ITST)*. 1–5.

[14] Robert R. Chodorek and Agnieszka Chodorek. 2017. MPTCP protocol misbehaviour in high-speed, uncongested network. *Journal of Marine Engineering & Technology* 16, 4 (2017), 248–256.

[15] Robert R. Chodorek and Agnieszka Chodorek. 2017. Performance Evaluation of MPTCP Transmission of Large Data Objects in Computing Cloud. In *Advances in Systems Science*, Jerzy Świątek and Jakub M. Tomczak (Eds.). Springer International Publishing, 131–141.

[16] Shuo Deng, Ravi Netravali, Anirudh Sivaraman, and Hari Balakrishnan. 2014. WiFi, LTE, or Both?: Measuring Multi-Homed Wireless Internet Performance. In *Proceedings of the 2014 Conference on IMC*. New York, NY, USA, 181–194.

[17] Mo Dong, Qingxi Li, Doron Zarchy, P. Brighten Godfrey, and Michael Schapira. 2015. PCC: Re-architecting Congestion Control for Consistent High Performance. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*. Oakland, CA, 395–408.

[18] Mo Dong, Tong Meng, Doron Zarchy, Engin Arslan, Yossi Gilad, Brighten Godfrey, and Michael Schapira. 2018. Vivace: Online-Learning Congestion Control. In *15th USENIX Symposium on NSDI 18*. USENIX Association.

[19] Esmaeil Fakhimi, Parisa Daneshjoo, Saeid Rezaei, Ali Akbar Movassagh, Ramin Karimi, and Yongrui Qin. 2018. Mptcp throughput enhancement by q-learning for mobile devices. In *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE, 1171–1176.

[20] S. Floyd and T. Henderson. 1999. The NewReno Modification to TCP's Fast Recovery Algorithm. (1999).

[21] Alan Ford, Costin Raiciu, Mark J. Handley, and Olivier Bonaventure. 2013. TCP Extensions for Multipath Operation with Multiple Addresses. RFC 6824. (Jan. 2013).

[22] Fa Fu, Xing Zhou, Thomas Dreibholz, Keying Wang, Feng Zhou, and Quan Gan. 2015. Performance comparison of congestion control strategies for multi-path TCP in the NORNET testbed. *2015 IEEE/CIC International Conference on Communications in China (ICCC)* (2015), 1–6.

[23] Sangtae Ha, Injong Rhee, and Lisong Xu. 2008. CUBIC: A New TCP-friendly High-speed TCP Variant. *SIGOPS Oper. Syst. Rev.* 42, 5 (July 2008), 64–74.

[24] Bo Han, Feng Qian, Shuai Hao, and Lusheng Ji. 2015. An Anatomy of Mobile Web Performance over Multipath TCP. In *Proceedings of the 11th ACM CoNEXT '15*. Article 5, 5:1–5:7 pages.

[25] Huaizhong Han, Srinivas Shakkottai, C. V. Hollot, R. Srikant, and Donald F. Towsley. 2006. Multi-path TCP: a joint congestion control and routing scheme to exploit path diversity in the internet. *IEEE/ACM Trans. Netw.* 16, 6 (2006), 1260–1271.

[26] Jiangping Han, Yitao Xing, Kaiping Xue, David SL Wei, Guoliang Xue, and Peilin Hong. 2020. Leveraging Coupled BBR and Adaptive Packet Scheduling to Boost MPTCP. *arXiv preprint arXiv:2002.06284* (2020).

[27] Jiangping Han, Kaiping Xue, Yitao Xing, Peilin Hong, and David SL Wei. 2019. Measurement and Redesign of BBR-based MPTCP. In *Proceedings of the ACM SIGCOMM 2019 Conference Posters and Demos*. 75–77.

[28] Elad Hazan. 2016. Introduction to Online Convex Optimization. *Foundations and TrendsÂő in Optimization* 2, 3-4 (2016), 157–325.

[29] B. Hesmans, G. Detal, S. Barre, R. Bauduin, and O. Bonaventure. 2015. SMAPP: Towards Smart Multipath TCP-enabled Applications. In *Proceedings of the 11th ACM CoNEXT '15*. Article 28, 28:1–28:7 pages.

[30] Michio Honda, Yoshifumi Nishida, Lars Eggert, Pasi Sarolahti, and Hideyuki Tokuda. May 2009. Multipath Congestion Control for Shared Bottleneck. Proc.PFLDNet workshop, 19–24.

[31] Apple Inc. 2019. Use Multipath TCP to create backup connections for iOS. https://support.apple.com/en-us/HT201373. (2019).

[32] Raj Jain. 1991. *The art of computer systems performance analysis - techniques for experimental design, measurement, simulation, and modeling.* Wiley. I–XXVII, 1–685 pages.

[33] Nathan Jay, Tomer Gilad, Nogah Frankel, Tong Meng, Brighten Godfrey, Michael Schapira, Jae Won Chung, Vikram Siwach, and Jamal Hadi Salim. 2018. A PCC-Vivace Kernel Module for Congestion Control. https://netdevconf.org/0x12/session.html?a-pcc-vivace-kernel-module-for-congestion-control. (2018).

[34] Nathan Jay, Noga Rotman, Brighten Godfrey, Michael Schapira, and Aviv Tamar. 2019. A deep reinforcement learning perspective on internet congestion control. In *International Conference on Machine Learning*. 3050–3059.

[35] Frank Kelly, Aman Maulloo, and David Tan. 1998. Rate control in communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society* 49 (1998), 237–252.

[36] Frank Kelly and Thomas Voice. 2005. Stability of End-to-end Algorithms for Joint Routing and Rate Control. *SIGCOMM Comput. Commun. Rev.* 35, 2 (April 2005), 5–12.

[37] Frank Kelly and Thomas Voice. 2005. Stability of End-to-end Algorithms for Joint Routing and Rate Control. *SIGCOMM Comput. Commun. Rev.* 35, 2 (April 2005), 5–12. https://doi.org/10.1145/1064413.1064415

[38] R. Khalili, N. Gast, M. Popovic, and J. Y. Le Boudec. 2013. MPTCP Is Not Pareto-Optimal: Performance Issues and a Possible Solution. *IEEE/ACM Transactions on Networking* 21, 5 (Oct 2013), 1651–1665.

[39] Ramin Khalili, Nicolas Gast, Miroslav Popovic, and Jean yves Le Boudec. 2013. Opportunistic Linked-Increases Congestion Control Algorithm for MPTCP. (2013).

[40] J. Kleinberg, Y. Rabani, and E. Tardos. 1999. Fairness in routing and load balancing. In *40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039)*. 568–578.

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

[42] Wenzhong Li, Han Zhang, Shaohua Gao, Chaojing Xue, Xiaoliang Wang, and Sanglu Lu. 2019. SmartCC: A Reinforcement Learning Approach for Multipath TCP Congestion Control in Heterogeneous Networks. *IEEE Journal on Selected Areas in Communications* 37, 11 (2019), 2621–2633.

[43] Yeon-sup Lim, Yung-Chih Chen, Erich M. Nahum, Don Towsley, Richard J. Gibbens, and Emmanuel Cecchet. 2015. Design, Implementation, and Evaluation of Energy-aware Multi-path TCP. In *Proceedings of the 11th ACM CoNEXT '15*. Article 30, 30:1–30:13 pages.

[44] Catalin Nicutar, Dragocs Niculescu, and Costin Raiciu. 2014. Using Cooperation for Low Power Low Latency Cellular Connectivity. In *Proceedings of the 10th ACM International on Conference on CoNEXT '14*. 337–348.

[45] Ana Nika, Yibo Zhu, Ning Ding, Abhilash Jindal, Y. Charlie Hu, Xia Zhou, Ben Y. Zhao, and Haitao Zheng. 2015. Energy and Performance of Smartphone Radio Bundling in Outdoor Environments. In *Proceedings of the 24th International Conference on WWW '15*. Republic and Canton of Geneva, Switzerland, 809–819.

[46] Wlodzimierz Ogryczak, Michal Pioro, and Artur Tomaszewski. 2005. Telecommunications network design and max-min optimization problems. (2005).

[47] Qiuyu Peng, Minghua Chen, Anwar Walid, and Steven Low. 2014. Energy Efficient Multipath TCP for Mobile Devices. In *Proceedings of the 15th ACM International Symposium on MobiHoc '14*. 257–266.

[48] Qiuyu Peng, Anwar Walid, Jaehyun Hwang, and Steven H. Low. 2016. Multipath TCP: Analysis, Design, and Implementation. *IEEE/ACM Trans. Netw.* 24, 1 (Feb. 2016), 596–609.

[49] Michal Pióro and Deepankar Medhi. 2004. *Routing, Flow, and Capacity Design in Communication and Computer Networks.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[50] B. Radunovic and J. Y. Le Boudec. 2007. A Unified Framework for Max-Min and Min-Max Fairness With Applications. *IEEE/ACM Transactions on Networking* 15, 5 (Oct 2007), 1073–1083.

[51] Costin Raiciu, Mark J. Handley, and Damon Wischik. 2011. Coupled Congestion Control for Multipath Transport Protocols. RFC 6356. (Oct. 2011). https://doi.org/10.17487/RFC6356

[52] Costin Raiciu, Mark J. Handley, and Damon Wischik. 2011. Coupled Congestion Control for Multipath Transport Protocols. RFC 6356. (oct 2011).
[53] Michael Schapira and Keith Winstein. 2017. Congestion-control throwdown. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*. 122–128.
[54] Intended Status St, Ards Track Q. Peng, J. Hwang, and S. Low. 2015. Balanced Linked Adaptation Congestion Control Algorithm for MPTCP. (2015).
[55] Meng Tong, Neta Rozen-Schiff, Brighten Godfrey, and Michael Schapira. 2020. Proteus: Scavenger Transport and Beyond. In *Proceedings of ACM SIGCOMM*.
[56] Brian White, Jay Lepreau, Leigh Stoller, Robert Ricci, Shashi Guruprasad, Mac Newbold, Mike Hibler, Chad Barb, and Abhijeet Joglekar. 2002. An Integrated Experimental Environment for Distributed Systems and Networks. Boston, MA, 255–270.
[57] Keith Winstein and Hari Balakrishnan. 2013. TCP Ex Machina: Computer-generated Congestion Control. *SIGCOMM Comput. Commun. Rev.* 43, 4 (Aug. 2013), 123–134.
[58] Damon Wischik, Costin Raiciu, Adam Greenhalgh, and Mark Handley. 2011. Design, Implementation and Evaluation of Congestion Control for Multipath TCP. In *Proceedings of the 8th USENIX Conference on NSDI'11*. 99–112.
[59] C. Xu, J. Zhao, and G. M. Muntean. 2016. Congestion Control Design for Multipath Transport Protocols: A Survey. *IEEE Communications Surveys Tutorials* 18, 4 (Fourthquarter 2016), 2948–2969.
[60] Zhiyuan Xu, Jian Tang, Chengxiang Yin, Yanzhi Wang, and Guoliang Xue. 2019. Experience-driven congestion control: When multi-path TCP meets deep reinforcement learning. *IEEE Journal on Selected Areas in Communications* 37, 6 (2019), 1325–1336.
[61] T. Zhu, X. Qin, L. Chen, X. Chen, and G. Wei. 2018. wBBR: A Bottleneck Estimation-Based Congestion Control for Multipath TCP. In *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*. 1–5. https://doi.org/10.1109/VTCFall.2018.8690919
[62] Martin Zinkevich. 2003. Online Convex Programming and Generalized Infinitesimal Gradient Ascent. (2003).

A PROOF SKETCH FOR THEOREM 4.1

We present below a sketch of the proof of Theorem 4.1. To simplify exposition, consider the scenario that the multipath connections have two subflows each and are purely loss-based, i.e., $\gamma = 0$. We later discuss how our arguments can be extended to the general case.

Proposition: In any equilibrium, all of a connection's subflows experience the same loss rates. Suppose (for point of contradiction) that in some equilibrium both of a connection's subflows send at non-zero rates and subflow 1 has a lower loss rate than subflow 2. Recall that the utility function penalizes the connection according to the maximum loss rate across subflows, i.e., subflow 2's loss rate. We show that, in this case, for a small enough $\epsilon > 0$, increasing subflow 1's rate by ϵ and decreasing subflow 2's rate by ϵ improves the connection's utility—a contradiction to this being an equilibrium. Intuitively, this is because the connection's total throughput remains unchanged yet the maximum loss rate across both flows is decreased when subflow's 2 rate is decreased[1].

Proposition: In any equilibrium, competing connections suffer the same loss rates. Consider two connections, i and j, with subflows sending at non-zero rates on the same link. Both connections' traffic on that link experience the same loss rate. This, together with the previous proposition, implies that all subflows of both connections experience the same loss rates.

Proposition: In any equilibrium, competing connections have the same total sending rate. Consider again two connections, i and j, with subflows sending at non-zero rates on the same link. By the previous propositions, both connections experience the same loss rates on all subflows. As this is an equilibrium, neither connection wishes to increase or decrease its rate. We observe that the above, combined with the structure of the connection-level utility function in (1) (specifically, its strict concavity in the presence of loss) implies that the two connections must have precisely the same total sending rate.

Proposition: Any equilibrium is LMMF. Consider the connection i with the lowest total sending rate in a certain equilibrium. Suppose, to simplify exposition, that only a single such connection exists. Observe that, by the previous proposition, that connection must be utilizing the entire link bandwidth on each of its subflows (as no other connection has the same total sending rate). This implies that connection i cannot possibly be assigned more bandwidth in any LMMF outcome, which, in turn, implies that in any LMMF outcome the worst-off connection cannot possibly be assigned more bandwidth than connection i and so the equilibrium is max-min-fair. By removing the links traversed by i's subflows from consideration and repeating this argument we can establish that the bandwidth assigned to the second-to-last worst-off connection is also maximized in the equilibrium, and so on.

Extending the above arguments to the general case. Observe that the above arguments did not rely on a connection having precisely two subflows and would hold for an arbitrary number of subflows. If connections are not purely loss based the same arguments apply with loss substituted with the aggregate penalty for loss and latency combined.

B PROOF SKETCH FOR THEOREM 5.1

We present below a sketch of the proof of Theorem 5.1. To simplify exposition, consider the scenario that the multipath connections are purely loss-based, i.e., $\gamma = 0$. We later discuss how our arguments can be extended to the general case.

Proposition: In any equilibrium, competing connections have the same total sending rate. Consider a global rate configuration that constitutes an equilibrium and two connections, i and j, with subflows sending at non-zero rates on the same link l. Let x_{il} and x_{jl} denote the sending rates of the subflow of connection i sending on link l and the subflow of connection j sending on link l, respectively. Let T_i and T_j denote the total sending rate across all subflows of connection i and connection j, respectively. Observe that as this is an equilibrium, link l must be fully utilized as otherwise each subflow on the link can achieve better utility by unilaterally increasing its sending rate—a contradiction to the equilibrium property. Suppose, for point of contradiction, that $T_i > T_j$ (the argument below also holds, by symmetry, for the case $T_j > T_i$). Plugging $L_r = \frac{C_r}{S_r}$ (as in [18]) and calculating the derivatives of the two subflows yields that the derivative of connection j's subflow is strictly higher. Intuitively, this follows from the fact that our per-subflow utility function (see Section 5.1) consists of two terms: a reward for increase in sending rate and a penalty for increase in loss rate. Since the two subflows are sending on the same link, an increase to the sending rate of Δ for one of them yields the same change to loss rate and so the same penalty. However, such an increase yields higher reward for connection j's subflow (since connection j's total sending rate is lower and $\alpha < 1$). However, as this is an equilibrium, the derivatives of both subflows must equal 0—a contradiction to $T_i > T_j$. We conclude that $T_i = T_j$.

Proposition: Any equilibrium is LMMF. The proof of this proposition is as in the identical proposition in the proof sketch for Theorem 4.1 in Appendix A.

[1]Our ability to identify such an ϵ is derived the connection-level utility functions being continuous and strictly concave in the appropriate region.

Extending the above arguments to the general case. If connections are not purely loss based the same arguments apply with loss substituted with the aggregate penalty for loss and latency combined.

C PROOF SKETCH FOR THEOREM 5.2

We present below a sketch of the proof of Theorem 5.2. We view the interaction of the different subflows of the different connections as a non-cooperative game in which each individual subflow is a player, the strategy of the subflow is its sending rate, and the subflow selfishly optimizes its local, subflow-specific utility function (as in MPCC). To simplify exposition, we focus below on the scenario that connections are purely loss-based. We then explain how our arguments can be extended to latency-sensitive connections.

Claim: Without loss of generality, no two subflows of the same connection share the same link. We observe that when two or more subflows of the same connection share the same link, the utility value of all these subflows at every point in time is identical (as it depends on the sum of sending rates across all of the connection's subflows and as all subflows on the same link experience the same loss rate and latency). Moreover, the same change in rate translates to the same change in utility function for all. Thus, w.l.o.g., all subflows belonging to the same connection that run on the same link will always reach the same decision. We thus henceforth assume that each of a connection's subflows runs on a different link.

Proposition: From some point in time onwards, every link in the network is fully utilized. Specifically, let S_r be the aggregate traffic across link r with capacity c_r. We show that from some point in time onwards S_r is always within the interval $(c_r, c_r \cdot (1 + \frac{1}{\beta-2})]$, where β is the coefficient in the subflow's utility function, as in 2. This follows from analyzing the gradient of the utility function of a subflow on that link. Trivially, if $S_r < c_r$ then the subflow can always benefit from increasing its rate, as for a small enough increment to the rate this can improve its throughput without inducing loss. We show that the derivative of the utility is always negative in the region if $S_r > c_r \cdot (1 + \frac{1}{\beta-2})$. Intuitively, this is because the linear gain in throughput is no longer worth the superlinear penalty for the increase in loss rate.

Proposition: From some point in time onwards, the loss rate on each link r is within the interval $(0, \frac{1}{\beta-1}]$. This proposition immediately follows from the previous proposition and the function for loss rate on a link r, $L_r = 1 - \frac{c_r}{S_r}$.

Proposition: When the subflows of two connections compete over the same link, the less the connection is sending in total the higher the derivative of its per-subflow utility for that link. Consider two connections, i and j, with subflows sending at non-zero rates on the same link l. Let the total sending rates of i and j across all their subflows be X_i and X_j, respectively, and suppose that $X_i < X_j$. The fact that the derivative of connection i's subflow is higher than that of connection j's subflow is immediately derived from the formulas of the derivatives $\frac{\partial U_i^{(l)}}{\partial X_i^l}$ and $\frac{U_j^{(l)}}{\partial X_j^l}$.

Iterative convergence. The proof now follows from the following observation. Consider the connection with the minimal goodput (sum of bandwidths utilized by this connection across all its subflows). Observe that when the goodput of this connection can only increase with time as, by the previous proposition, its utility-gradient is higher on all its links than than of each of its competitors. Thus, the minimum goodput across all connections will constantly increase until it stabilizes. Once this happens, the same arguments apply to the second-to-last goodput, etc. Once the rates stabilize, it holds for each connection and each of its links, that either the link is fully-utilized by the connection or the connection is only competing on the link with connections that have the same total sending rate. Arguments similar to those in Theorem 4.1 establish that this final state must be an LMMF equilibrium.

Extending the above arguments to the general case. If connections are not purely loss based the same arguments apply with loss substituted with the aggregate penalty for loss and latency combined.