Two Families of Optimal Multipath Congestion Control Protocols

Akshit Singhal, Xuan Wang, Zhijun Wang, Hao Che and Hong Jiang
Computer Science and Engineering, The University of Texas at Arlington, Arlington, U.S.A.
{akshit.singhal2, xuan.wang2}@mavs.uta.edu, hche@cse.uta.edu, {zhijun.wang, hong.jiang}@uta.edu

2022 IEEE 30th International Conference on Network Protocols (ICNP) | 978-1-6654-8234-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICNP55882.2022.9940376

***Abstract*—Multiple Path Transmission Control Protocols (MPTCPs) allow flows to explore path diversity of datacenter networks and multihoming to improve throughput, reliability, and network resource utilization. However, the existing MPTCPs are largely empirical by design and fall short of achieving satisfactory tradeoffs among responsiveness, TCP fairness and throughput. By leveraging the TCP utility and a network-utility-maximization (NUM) solution for concave utilities, in this paper, we derive and implement in Linux kernels two distinct families of NUM-optimal MPTCP protocols, EUTCP(γ), a function of a rate-scaling vector of the sub-flow rate-scaling coefficients, γ, and WUTCP(ω), a function of a utility weight vector of the sub-flow weights, ω, respectively. While the former allows resource pooling, the latter does not. We then show that the Semicoupled algorithm and EWTCP are in fact EUTCP(1) and WUTCP($1/m^2$), where m is the number of sub-flow paths, and hence, are NUM-optimal. The performance of the two families with equal weight and equal rate-scaling coefficient for all sub-flows when coexisting with TCP is also analyzed based on experiments in a testbed. In particular, the test results demonstrate that the family members of EUTCP(γ) with rate-scaling coefficient in the range of [1, 1.1] outperform three well-known MPTCPs with resource pooling capability, including LIA, OLIA and Balia, in terms of achieving satisfactory tradeoffs among responsiveness, fairness and throughput. Finally, the effectiveness of the proposed algorithms compared to the existing ones is further confirmed by simulation in a fat-tree datacenter network topology running both long and short flows.**

I. Introduction

Multiple Path Transmission Control Protocols (MPTCPs) split a flow into multiple subflows to be sent via different paths towards a destination exploring path diversity to improve flow throughput, network reliability and utilization for datacenter networks and multihomed users. Just like end-to-end TCP, most prevalent MPTCPs, such as LIA [1], OLIA [4] and Balia [3], are end to end, involving two endpoints only. This paper exclusively focuses on the design of end-to-end MPTCPs.

However, despite significant effort made in the past two decades in an attempt to develop optimal MPTCPs [6], the existing MPTCPs are largely empirical by design. In their seminal work on LIA [1], which is standardized by IETF [7], Wischik, et. al. acknowledge that due to the lack of theoretical underpinning, LIA can only be designed empirically. Khalili, et. al. [4] further show that LIA is not even Pareto optimal and propose a Pareto-optimal MPTCP, known as OLIA, which however, is again designed with no global optimization objective in mind. Peng, et. al. [3] classify and study major MPTCP algorithms with respect to some necessary conditions to achieve optimal traffic control in terms of network utility maximization (NUM), hence, taking another step towards the design of an optimal MPTCP. However, the algorithm proposed by the authors, known as Balia [3], that seeks to improve responsiveness and TCP friendliness over LIA and OLIA, is again, designed empirically. Interesting enough, the Semicoupled algorithm studied in [3] turns out to be NUM-optimal, as we shall show in this paper. Although EWTCP [2] also turns out to be NUM-optimal, as we shall also prove in this paper, it is again originally designed empirically without a global objective in mind.

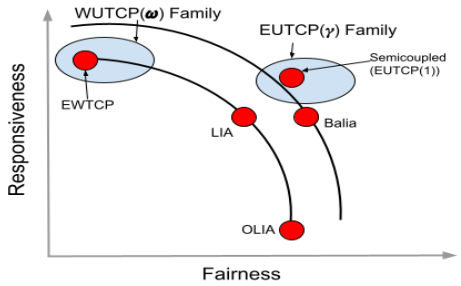


Figure 1: TCP Fairness vs Responsiveness

Another challenge facing the existing MPTCPs is that although most of them are designed with TCP friendliness in mind, they fall short of achieving satisfactory tradeoffs among responsiveness to network condition changes [9], TCP fairness and throughput. Fig. 1. illustrates the whereabouts for some well-known MPTCPs as well as EUTCP(γ) and WUTCP(ω) (to be defined shortly) in the responsiveness-and-TCP-fairness design space.[1] Without resource pooling, the WUTCP(ω) family including EWTCP (i.e., WUTCP($\{1/m^2\}$), as we shall show later, are highly responsive to network dynamics but cannot guarantee fairness with respect to TCP. Here resource pooling refers to the ability to dynamically allocate sub-flow rates based on the overall resource availability among all sub-flow paths. All the rest of MPTCPs allow resource pooling and hence can achieve better fairness with respect to TCP, at the cost of reduced responsiveness in general. As one can

This work was supported by the US NSF under Grant No. CCF SHF-2008835 and CCF SHF-2226117.

978-1-6654-8234-9/22/$31.00 ©2022 IEEE

[1]As we shall show, the throughput differences among different MPTCPs are relatively minor and hence is not considered here.

see, as a family of NUM-optimal MPTCPs, the EUTCP(γ) family including Semicoupled (i.e., EUTCP(**1**), as we shall prove later) covers a relatively large design space (i.e., the area covered by the oval) on the upper-right corner and hence, allowing better exploration of the design tradeoffs between responsiveness and TCP fairness than the existing ones.

This paper makes three major contributions. First by leveraging the NUM-optimal solution for concave utilities given in [13] and the concave TCP utility given in [12], for the first time, two families of NUM-optimal MPTCPs are derived, i.e., EUTCP(γ), based on a sub-flow rate-scaled TCP utility with rate scaling vector, $\boldsymbol{\gamma} = \{\gamma_1, ..., \gamma_m\}$ and WUTCP(ω), based on a weighted sum of sub-flow TCP utilities, with sub-flow weight vector, $\boldsymbol{\omega} = \{\omega_1, ..., \omega_m\}$, where m is the number of sub-flow paths. EUTCP(γ) allows resource pooling whereas WUTCP(ω) does not. With proper parameter settings, both families are TCP friendly by design because both are based on the TCP utility.

Second, we show that two empirically designed MPTCPs, i.e., the Semicoupled algorithm [3] and EWTCP [2] are in fact EUTCP(**1**) and WUTCP($\{1/m^2\}$), respectively. Namely they are NUM-optimal and family members of the NUM-optimal families derived in this paper.

Third, the two families are implemented in both Linux and NS3 simulator and their performance are tested against some well-known MPTCPs. Specifically, for EUTCP(γ), comparative performance analyses are carried out between its family members and some well-known resource-pooling-capable MPTCPs, including LIA [1], OLIA [4] and Balia [3]. The test results demonstrate that the family members in EUTCP($\boldsymbol{\gamma}$) with γ in the range of $[1, 1.1]$ outperform the other three MPTCPs in terms of responsiveness and fairness and is on par with the other three in terms of overall throughput performance. For WUTCP($\boldsymbol{\omega}$), we compare three family members at $\omega = 1/m^2, 1/m$, and $1/\sqrt{m}$. We conclude that for the TCP fairness criterion defined in [2], $\omega = 1/m$ should be used, instead of $\omega = 1/m^2$, as suggested in [2]. Finally, the effectiveness of the proposed algorithms compared to the existing ones is further verified by simulation in a fat-tree datacenter network.

II. BACKGROUND AND RELATED WORK

The existing MPTCPs can be broadly classified into two categories, with and without resource pooling capability.

For the category without resource pooling capability, a straightforward but naive solution is to simply run subflows as independent TCP flows. This approach, however, is too aggressive and unfair to single-path TCP flows, and without resource pooling, cannot balance the loads among subflow paths. This leads to the design of EWTCP [2], an equally weighted MPTCP. EWTCP attempts to achieve TCP fairness by modifying the previous solution, i.e., reducing the TCP window increase rate by a factor of $1/m^2$ for all m TCP-based subflows. However, besides the lack of load balancing capability inherited from the previous approach, EWTCP may lead to a flow rate lower than the best case single-path TCP.[2], discouraging users to use it.

For the other category, i.e., the one with resource pooling capability, there has been a great effort made in the last two decades in an attempt to develop MPTCP algorithms that are proven to be globally optimal in terms of network utility maximization (NUM), which in the form of a fluid-flow model, can be formally stated as follows:

$$max \sum_{i=1}^{n} U_i(x_{i,1}, x_{i,2}, ..., x_{i,m_i}), \tag{1}$$

subject to link bandwidth constraints,

$$\sum_{i,j:l \in L_{i,j}} x_{i,j} - c_l \leq 0; \qquad l \in L, \tag{2}$$

where n, m_i, L and $L_{i,j}$ are the number of active flows, the number of subflows in flow i, the set of links in the network, and the set of links that lie in the path of subflow j in flow i, respectively; c_l is the link bandwidth for link $l \in L$; and $U_i(x_{i,1}, x_{i,2}, ..., x_{i,m_i})$ is the user utility for flow i as a function of flow rates, $x_{i,j}$, for subflow j, $j = 1, 2, ..., m_i$.

The design goal is to find distributed solutions to the above NUM problem in the form of distributed flow rate control laws for individual subflows, using only binary congestion information feedback for control, i.e., whether a sub-flow/flow path is congested or not, which is essential to facilitate the development of end-to-end protocols including MPTCPs. Such control laws can then serve as the theoretical underpinning for the design of optimal rate or window-based MPTCP algorithms for any given user utilities that dictate the fairness criterion for resource allocation.

Kelly, et. al. [6] and Han, et. al. [22] convert the NUM problem into its Lagrange dual problem and then solves a relaxation of the dual problem by closely approximating it by incorporating a price function in the utility function. This approach is Pareto optimal, but it is not strictly NUM-optimal and it only selects one subflow path at a time and hence, suffers from flappiness and slow responsiveness [1]. Inspired by this approach, LIA [1], [7], [14] is proposed by modifying the previous approach to allow flow rate load balancing among multiple paths to avoid flappiness and improve response time, at the cost of losing the Pareto optimality. OLIA [4], an improved version of LIA, possesses the Pareto optimality. Using a duality model [17], [18], Balia [3] is developed to attempt to slove the above NUM. On the basis of the work in [17], [18], Peng, et. al. [3] further classify and study major MPTCP algorithms with respect to some necessary conditions to attain the NUM objectives, taking another step towards the design of an optimal MPTCP. However, the proposed Balia that seeks to improve responsiveness and TCP friendliness over LIA and OLIA, is again, heuristic by design. Polishchuk et. al. [15], [16] proposed mHIP, a TCP-friendly congestion control protocol for multipath host identity protocol to increase

[2]The best case single-path TCP is defined as the maximum flow rate the single-path TCP can achieve on any of the sub-flow paths available to MPTCP.

the resource utilization and improve fault tolerance, which however, is not an NUM-based solution.

Recently, Tabassum et.al. [24], [25] propose two more elaborate variations of LIA and OLIA to improve the throughput performance of LIA and OLIA. However, the solutions are complex and empirical, requiring dynamic adjustment of a decreasing parameter of the congestion window based on measured round trip time (RTT) and/or packet loss rate.

Meanwhile, in a series of works, Lagoa, et. al. [8], [10], [11], [13] directly solve a generalized version of the above NUM that permits sub-flow rate constraints by means of Sliding Mode Control in control theory [23]. The resulting control laws enable multiple classes of service and require only binary congestion information feedback for control, and hence are particularly suitable to serve as the theoretical underpinning for the design of MPTCPs. Wang, et. al. [12], [20] successfully apply this solution to reverse engineer TCP to arrive at the TCP utility function corresponding to the TCP Reno congestion control[3] and subsequently, design a TCP-friendly, end-to-end, single-path soft-minimum-rate-guaranteed congestion control protocol. Wang, et. al. [19] also successfully apply this solution to the development of an integrated congestion control and load balancing framework for datacenter networks, including a toy example demonstrating the viability of the solution for the development of MPTCP algorithms. Our work is motivated by these results. In particular, by leveraging the results given by Lagoa, et. al. [13] and Wang, et. al. [20], and with proper selection of user utility functions for multipath flows, we are able to derive two families of NUM-optimal MPTCPs, covering both MPTCP categories.

III. Two Families of NUM-Optimal MPTCPs

In this section, we first introduce the NUM-optimal multipath congestion control solution given in [10], the TCP utility function derived in [12], and then derive EUTCP($\boldsymbol{\gamma}$) and WUTCP($\boldsymbol{\omega}$), in separate subsections.

A. NUM-optimal multipath congestion control laws

According to [10], with respect to the NUM problem given in Eq. (1) (i.e., without minimum flow rate requirements), optimal control law for subflow j in flow i can be written as,

$$\dot{x}_{i,j} = z_{i,j}(t, x_{i,j}, cg_j)[f(x_{i,j}) - (1 - \overline{cg_j})] \tag{3}$$

with

$$f(x_{i,j}) = 1 - e^{-\partial U_i(x_{i,1}, x_{i,2}, ..., x_{i,m_i})/\partial x_{i,j}}, \tag{4}$$

where $U_i(x_{i,1}, x_{i,2}, ..., x_{i,m_i})$ can be any concave and strictly increasing function of $x_{i,j}$'s; $z_{i,j}(t, x_{i,j}, cg_j)$ can be any positive and piece-wise continuous scalar function and cg_j is the binary congestion indicator, cg_j =1 if the path the subflow j takes is congested and 0 otherwise; $\overline{cg_j}$ is the logical negation of cg_j. For a given concave utility function, a multipath congestion control law can be derived. For flows with non-concave user utility functions, the method proposed in [19] can be applied to derive the control laws.

[3]Note that this TCP utility function is the first one that captures both the slow start and congestion avoidance phases of TCP Reno.

B. Utility function of a single path TCP

It has long been recognized [5] that the utility function of logarithmic form leads to NUM-optimal control law that resembles the TCP behaviors in its congestion avoidance phase. This motivated the researchers to look for utility functions that can better match the TCP behaviors, notably, the ones given in [17] and [12]. While the one given in [17] is tied to a specific active queuing mechanism in the routers and hence is not end-to-end, the one given in TERSE [12] is end-to-end and by far the most accurate one that matches the TCP behaviors in both the slow start and congestion avoidance phases. Hence in this paper, we adopt the one in TERSE [12], which is described below.

Consider a fluid-flow version of the generic single path TCP congestion control algorithm with a slow start phase (SSP) and congestion avoidance phase (CAP), where αx is the multiplicative increase rate in SSP, μ is the additive increase rate in CAP, and βx is the multiplicative decrease rate in both SSP and CAP, meaning that retransmission timeout is treated the same way as three duplicated ACKs. Consequently, once the system enters CAP after the initial SSP, it will stay in CAP, implying that the congestion control algorithm in CAP will determine the steady state flow rate allocation. To limit the exposure, hereafter, we skip the subscription i for flow i. Then applying this generic TCP control algorithm to match Eq. (3), TCP utility, U_{tcp}, and the corresponding $z_{i,j}(t, x, cg)$ function can be reverse engineered and given as follows:
For the slow start phase (SSP)

$$U_{tcp}(x) = xlog(1 + \frac{\alpha}{\beta}) \tag{5}$$

and

$$z(t, x, cg) = (\alpha + \beta)x. \tag{6}$$

For the congestion avoidance phase (CAP)

$$U_{tcp}(x) = (\frac{\mu}{\beta} + x)[log(\mu + \beta x) - 1] - x[log(\beta x) - 1] \tag{7}$$

and

$$z(t, x) = \mu + \beta x \tag{8}$$

It can be easily verified that by plugging Eqs. (5), (6), (7), and (8) into (3) and (4), and by considering the fact that multiplicative increase rate is much larger than the additive increase rate, i.e., $\beta x_l \gg \mu$, and $\beta x \gg \mu$, we arrive at the following SSP and CAP control laws:

$$\dot{x} = \begin{cases} \alpha x & if \quad cg = 0 \\ -\beta x & if \quad cg = 1, \end{cases} \tag{9}$$

for SSP, and

$$\dot{x} = \begin{cases} \mu & if \quad cg = 0 \\ -\beta x & if \quad cg = 1 \end{cases} \tag{10}$$

for CAP.

These control laws are fluid-flow based, and can be approximately converted to window based control protocols. In the context of TCP Reno, which is window based, the flow rate is considered as a constant during each Round Trip Time (RTT), τ, and adjusted every RTT. Let W and ΔW be the congestion control window size and change of W at each RTT epoch, respectively. As the congestion window size is doubled or halved in SSP and increased by one MSS (i.e., the maximum segment size) or halved in CAP, without or with congestion, respectively, α, β and μ can be approximated as,

$$\alpha \approx 2\beta \approx 1/\tau, \qquad \mu = MSS/\tau \tag{11}$$

With these parameter settings, it can be easily shown that the SSP and CAP in Eqs. (9) and Eqs.(10) recover the TCP reno. Namely,

In SSP:

$$\Delta W = \begin{cases} W & \text{if } cg = 0 \\ -W/2 & \text{if } cg = 1. \end{cases} \tag{12}$$

In CAP:

$$\Delta W = \begin{cases} 1 & \text{if } cg = 0 \\ -W/2 & \text{if } cg = 1. \end{cases} \tag{13}$$

Finally, we note that α, β and μ in Eq. (11) are functions of RTT, τ. For a multipath flow, different subflows may see different RTTs, τ_i, for $i = 1, 2, ..., m$. So for multipath flows, we define τ to be the average RTT among all RTTs, i.e., $\tau = \sum_{i=1}^{m} \tau_i/m$.

C. The EUTCP(γ) Family

Utility function of an MPTCP flow: Theoretically, to ensure that a NUM-optimal MPTCP is friendly to TCP Reno by design, one can simply apply the TCP Reno utility function to the total flow rate of a multipath flow as follows,

$$U(x_1, x_2, ..., x_m) = U_{tcp}(\sum_{j=1}^{m} x_j), \tag{14}$$

In other words, the NUM-optimal solution is to equalize user utilities by balancing the sub-flow rates among sub-flow paths (hence, is resource pooling capable), which equalizes the rate allocation among both MPTCP and TCP flows, and hence, achieve TCP-friendly resource allocation.

However, our experiment results (see Table II) show that, just like LIA, OLIA and Balia, the resulting MPTCP corresponding to the above utility leads to skewed flow rate allocation with MPTCP flow rates higher than TCP flow rates. The reason is that, just like TCP, to allow sub-flow windows to increase when network resources become available, an MPTCP flow must maintain a positive minimum rate/window for each of its sub-flow, despite the fact that the optimal control law that underpins the MPTCP algorithm may require that the sub-flow rates be allowed to drop to zero when congestion occurs. Since TCP Reno uses 2×MSS as its minimum window size, most existing MPTCPs set the minimum window size for each sub-flow to be 2×MSS. This effectively makes the minimum window for the entire MPTCP flow to be $2m$×MSS, leading to skewed flow rate allocation in favor of MPTCP flows over TCP flows in practice, and the higher the number of sub-flows, m, the more skewed the flow rate allocation is.

One possible approach to remedying the above problem is to set the minimum window for each sub-flow to be one MSS instead of 2×MSS, as is the case for Balia. While our experiment shows that using one MSS in the MPTCP corresponding to the above utility can lead to almost perfect equal flow rate allocation in most cases, for some corner cases, it may cause unstable flow rate allocation. Namely, a sub-flow competing with TCP flows may not be able to grow its window back once the window reaches its minimum.

In this paper, we tackle the above challenge by using the following family of rate-scaled user utility functions instead,

$$U(x_1, x_2, ..., x_m) = U_{tcp}(\sum_{j=1}^{m} \gamma_j x_j), \tag{15}$$

where γ_j is a rate-scaling coefficient for sub-flow, j, for $j = 1, ..., m$. By setting some or all of the coefficients to be slightly larger than one, the NUM-optimal rate allocation that attempts to equalize user utilities is expected to allocate less rates to MPTCP flows than TCP flows, compensating for the skewed resource allocation.

EUTCP($\boldsymbol{\gamma}$): Let $z_l(t, x_l, cg_l)$ for each subflow l take the same format as its single-path counterpart given in Eqs. (6) and (8) with rate scaling, i.e.,

$$z_l(t, x_l, cg_l) = \begin{cases} (\alpha + \beta)\gamma_l x_l & \text{in SSP} \\ \mu + \beta\gamma_l x_l & \text{in CAP.} \end{cases} \tag{16}$$

Then substitute Eqs. (15) and (16) into Eqs. (3) and (4), we arrive at EUTCP($\boldsymbol{\gamma_l}$) as follows:

In SSP:

$$\dot{x_l} = \begin{cases} \alpha\gamma_l x_l & \text{if } cg = 0 \\ -\beta\gamma_l x_l & \text{if } cg = 1. \end{cases} \tag{17}$$

In CAP:

$$\dot{x_l} = \begin{cases} \frac{\mu + \beta\boldsymbol{\gamma_l} x_l}{\mu + \beta\gamma_l x}\mu & \text{if } cg = 0 \\ -\frac{\mu + \beta\gamma_l x_l}{\mu + \beta\gamma_l x}\beta\gamma_l x & \text{if } cg = 1. \end{cases} \tag{18}$$

Again, considering the fact that the multiplicative increase rate is much larger than the additive increase rate, i.e., $\beta x_l \gg \mu$, and $\beta x \gg \mu$, Eq.(18) can be approximated as

$$\dot{x_l} \approx \begin{cases} \frac{x_l}{x}\mu & \text{if } cg = 0 \\ -\beta\gamma_l x & \text{if } cg = 1. \end{cases} \tag{19}$$

EUTCP($\boldsymbol{\gamma_l}$) above simply states that the subflow increase rate is proportional to the ratio of the subflow rate and the overall flow rate, i.e., x_l/x, while its decrease rate is that of TCP Reno scaled by γ_l. It degenerates to TCP Reno at $m = 1$ and $\gamma = 1$.

Now we transform the above EUTCP($\boldsymbol{\gamma}$) to a window based one. Let $x = W \times MSS/\tau$ and $x_l = W_l \times MSS/\tau$. Then the congestion window size change for subflow l in each RTT, ΔW_l, according to Eqs. (17) and (19), are,

In SSP:

$$\Delta W_l \approx \begin{cases} \gamma_l W_l & \text{if } cg_l = 0 \\ -\frac{\gamma_l W_l}{2} & \text{if } cg_1 = 1. \end{cases} \tag{20}$$

In CAP:

$$\Delta W_l \approx \begin{cases} \frac{W_l}{\tau_l \sum_{j=1}^{m} W_j/\tau_j} & \text{if } cg = 0 \\ -\frac{\gamma_l W_l}{2} & \text{if } cg = 1. \end{cases} \tag{21}$$

D. The WUTCP($\boldsymbol{\omega}$) Family

Utility function of an MPTCP flow: Now we consider the following family of utility functions,

$$U(x_1, x_2, ..., x_m) = \sum_{j=1}^{m} \omega_j U_{tcp}(x_j), \tag{22}$$

where ω_j is the utility weight for sub-flow j. Clearly, this family of utility functions will lead to optimal rate allocation without resource pooling. For example, at $\omega_j = 1$ for $\forall j$, each sub-flow in a multipath flow is then treated the same way as a single path TCP flow, the most naive MPTCP solution in the category without resource pooling as mentioned in Section II.

WUTCP($\boldsymbol{\omega}$): Let,

$$z_l(t, x_l, cg_l) = \begin{cases} (\alpha + \beta)x_l & \text{in SSP} \\ \mu_l + \beta x_l & \text{in CAP.} \end{cases} \tag{23}$$

Here $\mu_l = MSS/\tau_l$ is the additive increase rate for subflow l. Similarly, we can easily get WUTCP($\boldsymbol{\omega}$) as follows:

In SSP, we have,

$$\dot{x_l} = \begin{cases} \frac{3}{2}(1 - 3^{-\omega_l})\alpha x_l & \text{if } cg_l = 0 \\ -3^{1-\omega_l}\beta x_l & \text{if } cg_l = 1. \end{cases} \tag{24}$$

In CAP, we have,

$$\dot{x_l} = \begin{cases} [1 - (\frac{\beta x_l}{\mu_l + \beta x_l})^{\omega_l}](\mu_l + \beta x_l) & \text{if } cg_l = 0 \\ -(\frac{\beta x_l}{\mu_l + \beta x_l})^{\omega_l}(\mu_l + \beta x_l) & \text{if } cg_l = 1. \end{cases} \tag{25}$$

Similarly, considering $\beta x_l \gg \mu_l$, then Eq. (25) can be approximated as

$$\dot{x_l} \approx \begin{cases} \omega_l \mu_l & \text{if } cg = 0 \\ -\beta x_l & \text{if } cg = 1. \end{cases} \tag{26}$$

From Eq. (26), we know that the rate increase for a subflow l is proportional to the utility weight ω_l. If $\omega_l < 1$, the subflow increase rate is smaller than that in a single path TCP flow (i.e.,$\omega_l \mu_l$ vs μ_l). If we set $\omega_l < 1$ for any subflow l, then each subflow obtains no more flow rate than that of a single path TCP in a shared link. WUTCP($\boldsymbol{\omega}$) can allocate more bandwidth to a multipath flow than the best case single path TCP in some cases, but it may also allocate less bandwidth to a multipath flow than the best case single path TCP in other cases, as we shall see later.

Now we convert WUTCP($\boldsymbol{\omega}$) into a window based one. In SSP, we have,

$$\Delta W_l = \begin{cases} \frac{3}{2}(1 - 3^{-\omega_l})W_l & \text{if } cg_l = 0 \\ -\frac{3^{1-\omega_l}}{2}W_l & \text{if } cg_l = 1, \end{cases} \tag{27}$$

and in CAP, we have,

$$\Delta W_l \approx \begin{cases} \omega_l & \text{if } cg = 0 \\ -\frac{W_l}{2} & \text{if } cg = 1. \end{cases} \tag{28}$$

Now we have derived two families of congestion control protocols, $EUTCP(\gamma)$ and $WUTCP(\omega)$. Different values of γ and ω result in different multipath congestion control protocols in their respective families.

IV. Performance Evaluation

In this section, we evaluate the performance of EUTCP($\boldsymbol{\gamma}$) and WUTCP($\boldsymbol{\omega}$) against some well-known MPTCPs, including EWTCP [2], LIA [1], OLIA [4] and Balia [3]. We first analyze these protocols, then we implement and perform experimental testing of them focusing on the fairness, responsiveness and throughput performance metrics. Finally, we test the performance of them in the presence of both short and long flows in a datacenter network by simulation.

A. Protocol Analysis

In the following protocol analysis, we only analyze CAP for all the MPTCPs for two reasons. First, the existing MPTCPs are mainly focused on the design and analysis of CAP, assuming that SSP for individual sub-flows follows that of TCP. Second, as aforementioned, the retransmission timeout is treated the same way as three duplicated ACKs in our solutions, meaning that once entering CAP after initial SSP, the system will stay in CAP and hence, the long-run flow rate allocation is determined by CAP only.

Table I lists the congestion control protocols in CAP for all the MPTCPs to be studied in this section. The protocols are classified into two distinct categories, i.e., with or without resource pooling capability, are listed separately. Note that the increasing part of each protocol is given on a per ACK bases not per RTT. As a result, the per-RTT-based window increase formula in Eq. (21) and Eq. (28) must be divided by W_l to arrive at the corresponding increase parts in Table I.

For the category with resource pooling capability, we first observe that the Semicoupled protocol is indeed a family member of EUTCP($\boldsymbol{\gamma}$), i.e., EUTCP ($\{\mathbf{1}\}$), meaning that it is NUM-optimal, achieving the global objective given in Eq. (15) at $\gamma_l = 1$, $\forall l$. Second, we note that, compared with all the other protocols in the category, the EUTCP($\boldsymbol{\gamma}$) family including the Semicoupled are the least complex and hence, most computationally efficient ones. This also implies that if some other solutions also turn out to be NUM optimal, the corresponding utility functions are likely to be substantially

Table I: Congestion control protocols in CAP

Solution	Increase (per Ack)	Decrease (per loss)	Comments
With Resource Pooling			
LIA (Coupled) [1], [6], [21]	$\frac{(W_l/\tau_l^2)}{(\Sigma_{i\epsilon R} W_i/\tau_i)^2}$	$\frac{W_l}{2}$	
Semicoupled [3] (optimal)	$\frac{1}{\tau_l(\Sigma_{i\epsilon R} W_i/\tau_i)}$	$\frac{W_l}{2}$	
OLIA [4]	$\frac{(W_l/\tau_l^2)}{(\Sigma_{i\epsilon R} W_i/\tau_i)^2} + \frac{\delta}{W_l}$	$\frac{W_l}{2}$	δ is modulation factor
Balia [3]	$\left\{\frac{(W_l/\tau_l^2)}{(\Sigma_{i\epsilon R} W_i/\tau_i)^2}\right\}(\frac{1+\alpha_l}{2})(\frac{4+\alpha_l}{5})$	$\frac{W_l}{2} min\{\alpha_l, \frac{3}{2}\}$	$\alpha_l \triangleq \underset{j\epsilon R}{max}\{x_j\}/x_l$
EUTCP($\boldsymbol{\gamma}$) (optimal)	$\frac{1}{\tau_l(\Sigma_{i\epsilon R} W_i/\tau_i)}$	$\frac{\gamma_l W_l}{2}$	γ_l is the scaling coefficient
Without Resource Pooling			
EWTCP [2] (optimal)	$\frac{\alpha}{W_l}$	$\frac{W_l}{2}$	α is weight for each route
WUTCP($\boldsymbol{\omega}$) (optimal)	$\frac{\omega_l}{W_l}$	$\frac{W_l}{2}$	ω_l is the subflow utility weight

more complex and hence, harder to interpret, particularly with respect to the fairness to TCP.

For the category without resource pooling, it becomes clear that EWTCP is indeed a family member of WUTCP($\boldsymbol{\omega}$) with $\omega_l = \alpha, \forall l$, and hence, is NUM optimal as well. In fact, in the original paper on EWTCP [2], it is suggested that $\alpha = 1/m^2$ should be used. By doing so, the paper shows that each sub-flow of an EWTCP flow sharing a bottleneck link with a TCP flow will then be allocated one mth of the TCP flow rate and hence is TCP fair, in the sense that the overall EWTCP flow rate is equal to the TCP flow rate if each and every sub-flow shares a bottleneck link with a TCP flow. The proof, however, is based on a TCP throughput model given in [27] that assumes that the retransmission timeout is a more frequent event than the three duplicated ACKs, and the TCP goes back to slow start phase after the retransimission timeout happens.

In fact, since we now know that EWTCP is NUM-optimal and its utility function is given by Eq. (22) with $\omega_l = \alpha$ for $l = 1, ..., m$, we can prove that to satisfy the above TCP fairness criterion, α should be set at $1/m$, instead of $1/m^2$. First, we note that since $\beta x \gg \mu$, U_{tcp} in Eq. (22) can be approximately written as $U_{tcp} \approx \frac{\mu}{\beta} log(\beta x)$, i.e., a log function of x. Then it can be easily shown [19] that for any given number of sub-flows, each having the weighted utility, αU_{tcp}, which share a bottleneck link with any given number of TCP flows, the sum of the utilities for all the flows/sub-flows sharing this link is maximized, if each sub-flow is allocated α times of the rate allocated to each TCP flow. This means that to meet the above TCP fairness criterion for EWTCP, α should be $1/m$, not $1/m^2$. This is also confirmed by the experiment results presented in the following section.

Nevertheless, the work in this paper is not meant to give a definitive answer as to what fairness criteria and hence, what parameters, $\boldsymbol{\omega}$ and $\boldsymbol{\gamma}$, or equivalently, which family members of the two families, should be adopted in practice. Instead, the objective of this work is to reveal the performance tradeoffs among the members of the two families as well as the other MPTCPs in the list so that users can make an informed decision as to which MPTCP should be adopted to best serve their application needs. For example, in a multi-homing scenario, different paths may charge different usage fees and/or offer different service qualities. In this case, a user may want to assign different ω_l's or γ_l's for different sub-flows to explore the tradeoffs between the performance and cost.

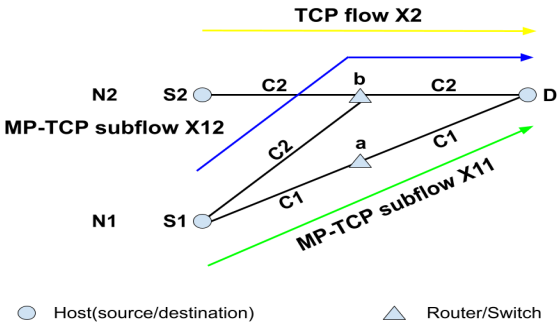


Figure 2: The network topology used in Linux implementation

B. Testbed Testing Result Analysis

For the ease of comparison with Balia [3], the state-of-the-art solution, we adopt the same network topology as the one used in [3]. Namely, all the test cases are performed based on the topology shown in Fig. 2. It involves two source hosts, S_1, sending N_1 MPTCP flows with two sub-flows (i.e., $m = 2$), x_{11} and x_{12}, and S_2, sending N_2 TCP flows, to the same destination host D. Nodes, b and a, provide a single path from S_1 to D via b and two sub-flow paths from S_2 to D, via a (i.e., sub-flow x_{12}) and b (i.e., sub-flow x_{11}), respectively.

The hosts (i.e., S_1, S_2 and D) are Dell Poweredge servers, each equipped with 8-core processors with 10GB memory and running Linux 16.04. Nodes a and b are Dell N4032F switches, each with multiple 1 Gbps Ethernet interfaces running Ubuntu 16.04.1 LTS (Linux kernel 4.19.98). The link bandwidth for all the links can be configured at any rate lower

than or equal to 1 Gbps through the networking interface traffic control command tc, allowing for the testing of the MPTCP responsiveness to sudden link bandwidth changes. Both MPTCP families are implemented by modifying the open source Linux kernel codes of LIA and Balia [26]. For both the EUTCP($\boldsymbol{\gamma}$) and WUTCP($\boldsymbol{\omega}$) families, we only test the single parameter cases, i.e., the cases where $\gamma_l = \gamma$ and $\omega_l = \omega$, $\forall l$. Hence, they can be simply written in terms of a single parameter, i.e., EUTCP(γ) and WUTCP(ω). We present the testbed results for the two families and the related MPTCPs in the two categories, separately, focusing on the fairness, throughput and responsiveness performance metrics.

*1) **EUTCP(γ):*** In this section, we study the performance of three family members, EUTCP(1), EUTCP(1.05) and EUTCP(1.1), together with LIA, OLIA and Balia in the testbed.

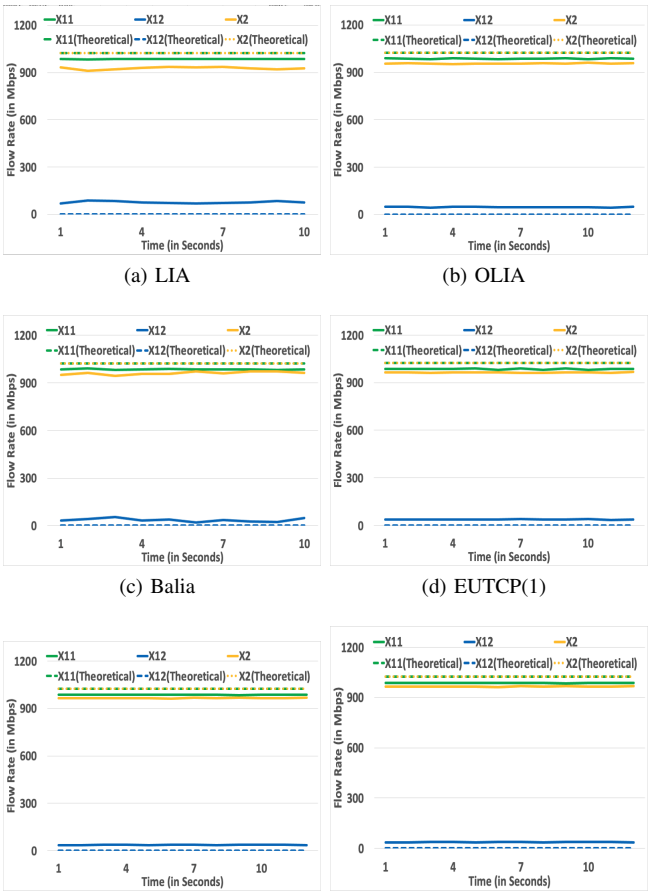
TCP Fairness and Throughput: First we test the performance of these MPTCPs in a balanced network by setting $C_1 = C_2 = 1024$ Mbps (i.e. 1 Gbps link) and $N_1 = N_2 = 1$, i.e., one multipath flow and one TCP flow (the TCP Reno in the Linux kernel is applied without modification). With this setup, all the MPTCPs will strive to equalize and maximize the flow rates for the two flows, i.e., all targeting at the optimal flow rate allocation: $x_1 = x_2 = 1024$ Mbps, $x_{11} = 1024$ Mbps and $x_{12} = 0$ Mbps.

In this experiment, we consider the protocol performance in the steady state. The two flows are long lived, meaning that each flow has unlimited amount of data to send and hence, lasts throughout the entire measurement window. The performance of EUTCP(1), EUTCP(1.05) and EUTCP(1.1) against with LIA, OLIA, Balia, are shown in 3 (a)-(f), respectively.

First, we note that for all MPTCPs, x_{11} (green) and x_2 (yellow) are below their respective optimal flow rate targets, i.e., the link bandwidths, whereas x_{12} (blue) are above its optimal flow rate target that is zero. This is inevitable. The former is due to the discrete-time (once every RTT) window-based adaptive flow control with delay, which guarantees that the achievable flow rate cannot saturate the link bandwidth. The latter is caused by the need to set a non-zero minimum window for a sub-flow as explained earlier, which prevents it from being allocated zero bandwidth. This also contributes to the reduction of the flow rate, x_2. Moreover, all the MPTCPs are able to achieve almost the same flow rate allocation for x_{11}. This is easy to understand as the entire sub-flow path is dedicated to this sub-flow without congestion.

Second, we note that in terms of flow rate allocation for x_2 and x_{12}, LIA offers the lowest rates, OLIA and Balia perform almost equally well, but not as good as the three members in the EUTCP(γ) family, all of which perform equally well.

In our next experiment, we study the average throughput and TCP fairness performance for long-lived MPTCP flows in steady state. The experiment setup is the same as the previous one, except now we have, N_1 = N_2 = 5. To quantify the TCP fairness, we define the TCP unfairness as in Eq. (29). The smaller the TCP unfairness is, the fairer the MPTCP is with



(a) LIA (b) OLIA

(c) Balia (d) EUTCP(1)

(e) EUTCP(1.05) (f) EUTCP(1.1)

Figure 3: Steady state performance of the EUTCP(γ) family

respect to TCP.

$$TCP\ Unfairness = \frac{TP_{MPTCP} - TP_{TCP}}{TP_{TCP}} \times 100 \quad (29)$$

Here, TP_{MPTCP} and TP_{TCP} are the per MPTCP and per TCP flow throughput, respectively. For each MPTCP, we repeat the experiment three times and take the average throughput. The results are shown in Table II, including average flow throughput for both multipath flows and TCP flows, the TCP unfairness, and the aggregate throughput, i.e., the throughput for TCP and multipath flows combined. As one can see, LIA is the least fair to TCP with TCP unfairness of 14.56%. OLIA and Balia significantly improve the TCP fairness over LIA, with TCP unfairness of 8.11% and 6.29%, respectively, at the cost of almost negligible deduction of the aggregate throughput, consistent with the results given in [3].

Moreover, EUTCP(γ) reaches the smallest TCP unfairness at about $\gamma = 1.05$ and then grows back up as γ further increases, as evidenced by the results for EUTCP(1.05) and EUTCP(1.1). The reason is that increasing γ from one makes the sub-flow window for x_{12} drop faster as the congestion occurs (see the decreasing part of the algorithm in Table I), reducing the skewed flow rate allocation and hence, improving TCP fairness. However, as the sub-flow window

Table II: Average flow throughput (Mbps), TCP unfairness, and aggregate throughput (Multiple flows of MPTCP flows and single path TCP flows combined) (Mbps) at $N_1 = N_2 = 5$.

	LIA	OLIA	Balia	EUTCP(1)	EUTCP(1.05)	EUTCP(1.1)
Average MPTCP throughput	1061.92	1032.78	1022.01	1023.54	1022.07	1034.19
Average TCP throughput	926.99	955.33	961.48	963.92	965.69	953.87
unfairness (%)	14.56	8.11	6.29	6.18	5.83	8.42
Aggregate Throughput	1988.91	1988.11	1983.49	1987.46	1987.75	1988.06

Table III: Convergence times (seconds) of MPTCP flows during bandwidth changes

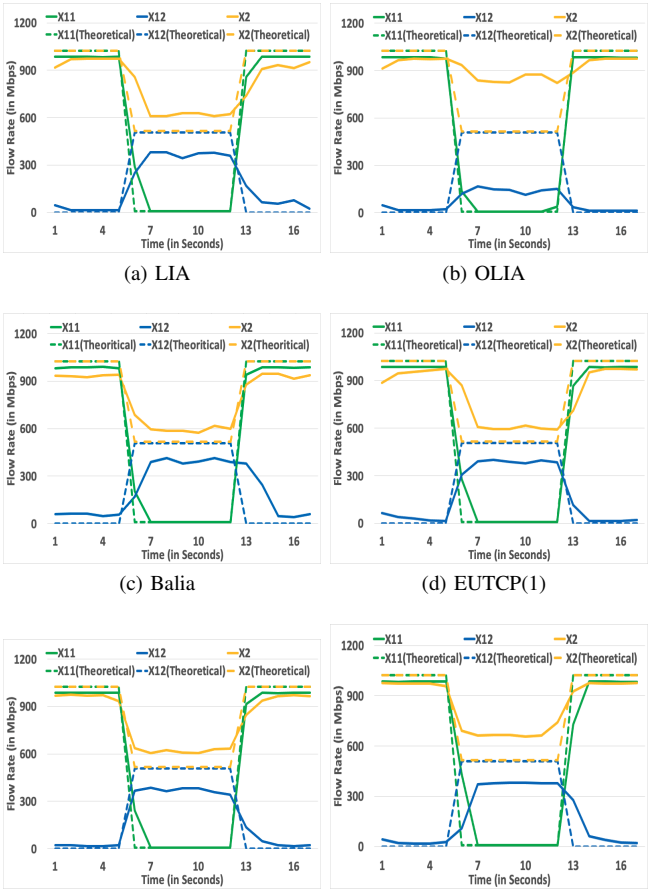
	LIA	Balia	EUTCP(1)	EUTCP(1.05)	EUTCP(1.1)
x_{11}	2	2	2	2	2
x_{12}	2.5	2.5	2	1.5	2

decreases faster, it also increases faster in the absence of congestion, simply because the sub-flow window increase rate is inversely proportional to the sum of sub-flow window sizes (see the increasing part of the protocol in Table I). These two competing effects result in the existence of a γ value around 1.05, where the TCP unfairness is minimized, i.e., about 5.83%, the smallest among all MPTCPs studied, without scarifying the throughput performance. In fact, the differences of the aggregate throughput among all the MPTCPs in Table II are extremely small and can be largely neglected. Furthermore, we note that EUTCP(1.05) improves EUTCP(1) in the TCP fairness performance by about 6%, implying that EUTCP(γ) is not too sensitive to γ and setting γ anywhere close to 1.05 should be good.

Responsiveness: Next, we test the performance of these MPTCPs in terms of responsiveness in a dynamically changing environment. To this end, we consider $N_1 = N_2 = 1$ and set the link bandwidth $C_1 = C_2 = 1024$ Mbps in the initial first 5 seconds; then suddenly change C_2 from 1024 Mbps to $C_2 = 8$ Mbps for the next 7 seconds (i.e., from second 6 to second 12); and finally switch it back to $C_2 = 1024$ Mbps. With this setup, all the MPTCPs will strive to arrive at the following flow rate allocation that equalizes and maximizes the flow rates for the two flows: $x_{11} = x_2 = 1024$ Mbps and $x_{12} = 0$ Mbps, before the 6th second and after 12th second, and $x_{11} = 8$ Mbps, $x_{12} = 508$ Mbps and $x_2 = 516$ Mbps from the 6th second to 12th second.

The throughput of the flows is given in Fig. 4 and the flow rate convergence times for all the MPTCPs except OLIA upon the bandwidth changes are given in Table III. First, we note that among all of them, OLIA is the least responsive, even though it outperforms LIA in terms of TCP fairness. It does not even come close to the new optimal flow rate allocation, compared with all the others, which is the reason why it is excluded from Table III for convergence comparison. LIA, Balia, EUTCP(1.1) and EUTCP(1) (or semicoupled algorithm) are on par with one another. Clearly, EUTCP(1.05) performs the best among all of them with the smallest convergence time.

However, one may notice that for all the MPTCPs, x_{12} (x_2) converges to a flow rate lower (higher) than the optimal one, similar to the experimental results given in [3]. We find that this skewed flow rate allocation in favor of the single-



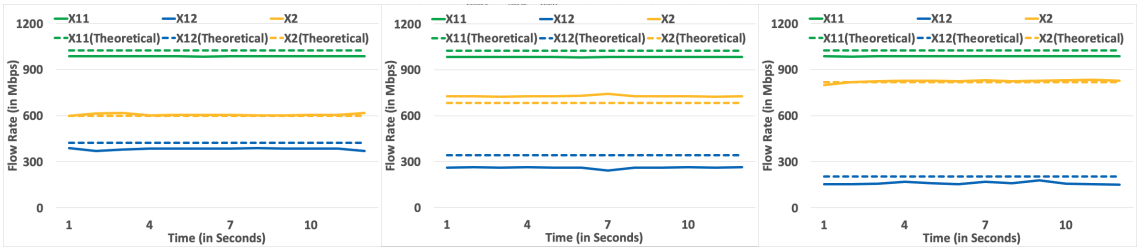
(a) LIA (b) OLIA

(c) Balia (d) EUTCP(1)

(e) EUTCP(1.05) (f) EUTCP(1.1)

Figure 4: Responsiveness with dynamic link bandwidths for MPTCPs in the resource-pooling-capable category

path TCP is largely caused by the way such protocols are normally implemented. A TCP flow, whether it is single path or multipath, is normally run by a single thread. As a result, the processing resource allocated to the window control for each sub-flow in a multipath flow reduces and hence, the processing delay for each sub-flow increases, as the number of sub-flow paths increases. For a sub-flow shares its sub-flow path with a TCP flow (e.g., x_{12} and x_2 in our case), it sees a larger RTT than the TCP flow as it incurs larger processing delay, hence, receiving lower flow rate allocation. This is particularly problematic in an experimental environment of ours, where the entire testbed is hosted in a single rack, where the end-to-end propagation delay is negligible, making the RTT and hence, the performance highly sensitive to the processing delay. The following experiment results will further confirm that this is indeed the case. A possible way to fix this problem is to



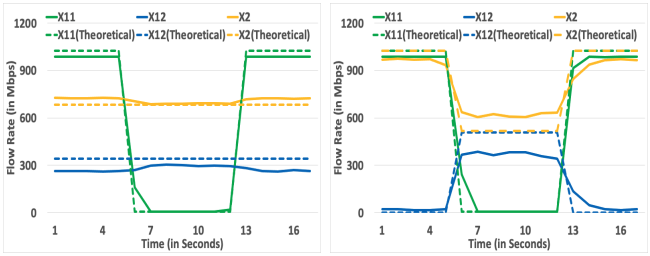
(a) WUTCP($1/\sqrt{m}$) (b) WUTCP($1/m$) (c) WUTCP($1/m^2$)

Figure 5: Performance of WUTCP(ω) Family.

assign more processing resources to MPTCP flows than TCP flows. How much more should be assigned to it, however, is an implementation issue to be addressed in the future.

Based on the above performance analysis, we conclude that the family members in EUTCP(γ) with γ taking values in the range of $[1, 1.1]$ offer the best tradeoffs among TCP fairness, responsiveness and throughput in the resource-pooling-capable category.

*2) **WUTCP(ω):*** Now we study the performance of WUTCP(ω) family in terms of TCP fairness, throughput, and responsiveness. We choose $\omega = 1/m^k$ with $k = 1/2$, 1 and 2, to study the impact of weights on the flow rate allocation.



(a) WUTCP(1/m) (b) EUTCP(1.05)

Figure 6: Responsiveness of WUTCP($\frac{1}{m}$) vs EUTCP(1.05) with dynamic link bandwidth.

TCP fairness and Throughput: Again, we first consider the following network setup: $C_1 = C_2 = 1024$ Mbps and $N_1 = N_2 = 1$. As aforementioned, the optimal rate allocation for a WUTCP($\boldsymbol{\omega}$) sub-flow is approximately ω times that of a TCP flow sharing the same bottleneck link. Hence with this setup, the optimal flow rate allocation is: $x_{11} = 1024$ Mbps, $x_{12} = 1024 \times \frac{\omega}{1+\omega}$ Mbps and $x_2 = 1024 \times \frac{1}{1+\omega}$ Mbps.

Fig. 5(a), (b) and (c) plot the results for the EUTCP(ω) members at $\omega = \frac{1}{\sqrt{m}}$, $\omega = \frac{1}{m}$ and $\omega = \frac{1}{m^2}$, respectively. Again, without experiencing any congestion, the sub-flow rate, x_{11} (green), is almost the same for all the three cases. Also for all the cases, the flow rate allocated to, x_2 (x_{12}), is always slightly higher (lower) than the corresponding optimal one, for the same reason discussed in the previous case. The results clearly demonstrate that the case with $\omega = 1/m = 1/2$ indeed gives flow rate allocation much closer to the desired ratio, $\frac{x_{12}}{x_2} = \frac{1}{2}$ than the case with $\omega = 1/m^2 = 1/4$, which instead gives the ratio, $\frac{x_{12}}{x_2} = \frac{1}{4}$ with pretty high accuracy as our model predicts. These results further support our earlier claim that to satisfy the TCP fairness criterion given in the paper on EWTCP [2], the weight α should be set at $\alpha = \frac{1}{m}$, not $\frac{1}{m^2}$.

Responsiveness: As discussed above, MPTCPs without resource pooling capability, including WUTCP(ω), cannot respond to network dynamics effectively, especially in terms of maintaining TCP fairness. In what follows, we only give the experiment results on WUTCP($\frac{1}{m}$) to demonstrate this.

Consider the same experiment setup as the previous responsiveness testing case for EUTCP(γ). The only difference is that now the MPTCP flow, x_1, is run by WUTCP($\frac{1}{m}$).

As shown in Fig. 6 (a), as C_1 drops from 1024 Mbps to 8 Mbps, x_{12} and x_2 change slightly. In fact, in theory, they should not change at all because without resource pooling capability, EUTCP(γ) controls the send windows for individual sub-flows independently, as evidenced by the EUTCP(γ) algorithm given in Table I. The fact that both x_{12} and x_2 moved closer to their respective optimal values during the time period where $C_1 = 8$ Mbps further confirms that the actual processing resources allocated to individual sub-flows have an impact on the flow rate allocation. When C_1 drops from 1024 Mbps to 8 Mbps, the transmission delay for packets in the sub-flow x_{11} increases substantially, resulting in more than $10\times$ increase of its RTT, as the measured statistics show. This means that the processing resource demand for sub-flow, x_{11}, is also reduced by that many times, yielding much of the processing resource to sub-flow, x_{12}. This effectively reduces the RTT for x_{12} and hence, the competitiveness of x_{12} with respect to TCP flow, x_2, resulting in reduced skew for flow rate allocation during the time period when $C_1 = 8$ Mbps. Similar behaviors are also observed for other family members in EUTCP(γ). This case study also confirms the earlier claim that MPTCP without resource pooling may perform worse than the best case single path TCP.

In contrast, as plotted in Fig. 6 (b) again for the case of EUTCP(1.05), the resource pooling capability of EUTCP(1.05) helps to rebalance the flow rate allocation to maintain equal share of the flow rate allocation, or to meet the same fairness criterion, in response to C_1 changes.

C. Performance Analysis by Simulation

So far we have focused on the performance evaluation of the two families of the NUM-optimal MPTCPs in terms of

Table IV: Performance Comparison in Datacenter Network

Solutions	**Short Flow Finish Time (mean/stdev)**	**Completed Short Flows**	**Long Flow Goodput (mean/stdev)**	**Completed Long Flows**	**Core Layer Utilization (mean)**
MMPTCP	121.502/± 139.077 ms	5265	59.129/± 17.473 Mbps	168	71.329 %
LIA	85.422/± 98.638 ms	4961	61.665/± 18.590 Mbps	168	74.871 %
EUTCP(1)	102.48/± 143.722 ms	5085	61.472/± 19.109 Mbps	168	75.461 %
EUTCP(1.05)	105.008/± 159.195 ms	5113	61.775/± 18.994 Mbps	168	75.572 %
EUTCP(1.1)	103.475/± 150.486 ms	4953	61.842/± 19.170 Mbps	168	75.576 %
Balia	85.550/± 78.843 ms	5023	62.146/± 18.574 Mbps	168	75.773 %
WUTCP($1/m^2$)	80.269/± 82.717	4993	62.065/± 18.829 Mbps	168	75.612 %
WUTCP($1/m$)	106.75/± 155.709	4988	61.780/± 19.055 Mbps	168	75.764 %
WUTCP($1/\sqrt{m}$)	85.55/± 78.843	5066	59.429/± 19.711 Mbps	168	73.679 %

responsiveness, fairness and throughput for long flows in a small testbed. In this section, we focus on the performance evaluation of the two families in supporting a mix of short and long flows in a datacenter network by simulation. The performance of the two families are compared against MMPTCP [28], in addition to LIA and BALIA. Unlike other MPTCPs that behave much like TCP in the slow start phase, MMPTCP randomizes the packet distribution to sub-flow paths in the slow start phase, aiming at reducing the short flow completion time. Without an open source Linux kernel implementation, we did not compare against this solution in the testbed.

Here we use the same simulation setup and the open source code as that in [28] to ensure that MMPTCP achieves the intended performance as that in the original paper. The datacenter network is a 4:1 oversubscribed FatTree topology consisting of 512 servers. Each MPTCP flow has eight sub-flows. One third of the servers run long (background) flows and the rest run short flows (70KB each) which are scheduled by a central scheduler following the Poisson arrival process.

The results are presented in Table IV. We observe that even though LIA and BALIA provide smaller short flow completion times, compared with the other MPTCPs with resource pooling, they do not perform well in terms of the number of completed flows. This is because there are some short flows that don't finish due to frequent timeouts or retransmission. On the other hand, as expected, MMPTCP offers the highest number of short flows completion, 5265 to be exact, but it suffers from the worst performance in terms of long flow goodput among all MPTCPs with resource pooling.

It is clear that EUTCP(1) and EUTCP(1.05) outperform MMPTCP in terms of the short flow finish times. In the meantime, EUTCP(1.05) attains a high number of short flow completions that is within 3% of that for MMPTCP. Moreover, it also achieves relatively high performance in terms of both goodput for long flows and core layer utilization, compared with the other MPTCPs. Therefore, it strikes the best balance among all MPTCPs. Note that even though WUTCP family also perform well, they are inherently incapable of resource pooling and hence, cannot effectively respond to network condition changes.

Based on the above performance analysis, we summarize the performance in terms of fairness and responsiveness on long flow workload and mixed workload in Table V. Note that the aggregate throughputs for different MPTCPs are close to one another as shown in Table II and hence the throughput is not included as a performance metric in the table. From the summary given in this table, we conclude that EUTCP(γ) with γ in the range of $[1, 1.1]$ offer the best overall performance among all the MPTCPs tested in this paper.

Table V: Performance comparison of MPTCPs (poor:1-3, fair/medium: 4-6 and good/fast: 7-10).

Solutions	**TCP fairness**	**Responsiveness**	**Mixed Workload**
With Resource Pooling			
LIA (Coupled)	poor(3)	medium(6)	fair(6)
OLIA	fair(6)	poor(1)	N.A.
Balia	good(8)	medium(6)	good(7)
EUTCP(1)	good(8)	medium(8)	good(7)
EUTCP(1.05)	good(10)	fast(10)	good(9)
EUTCP(1.1)	fair(6)	fast(8)	fair(6)
Without Resource Pooling			
EWTCP or WUTCP($\frac{1}{m^2}$)	poor(1)	fast(10)	good(7)
WUTCP($\frac{1}{m}$)	fair(4)	fast(10)	good(7)

V. Conclusions

In this paper, we derive and implement in Linux two distinct families of Network Utility Maximization (NUM)-optimal Multiple Path Transmission Control Protocol (MPTCP) protocols, EUTCP($\boldsymbol{\gamma}$) and WUTCP($\boldsymbol{\omega}$), by leveraging the TCP utility function and the NUM solution for concave utilities. EUTCP($\boldsymbol{\gamma}$) is a function of a rate-scaling vector of the sub-flow rate-scaling coefficients, $\boldsymbol{\gamma}$ and WUTCP($\boldsymbol{\omega}$) is a function of a utility weight vector of the sub-flow weights, $\boldsymbol{\omega}$. We also show that the Semicoupled algorithm is a protocol in the family of EUTCP($\boldsymbol{\gamma}$) with $\boldsymbol{\gamma} = 1$ and EWTCP is a protocol in the family of WUTCP($\boldsymbol{\omega}$) with $\boldsymbol{\omega} = 1/m^2$, where m is the number of sub-flow paths, and hence, are NUM-optimal. The performance of the two families is also analyzed based on experiment in a testbed and simulations on a datacenter FatTree topology. The test results demonstrate that the family members with equal sub-flow rate-scaling coefficient setting in the range of $[1, 1.1]$ in EUTCP($\boldsymbol{\gamma}$) outperform three well-known MPTCPs with resource pooling capability, including LIA, OLIA and Balia, in terms of responsiveness and fairness and are on par with the three MPTCPs in terms of the throughput performance.

REFERENCES

[1] Wischik Damon, Costin Raiciu, Adam Greenhalgh, and Mark Handley. "Design, Implementation and Evaluation of Congestion Control for Multipath TCP." In NSDI, vol. 11, pp. 8-8. 2011.

[2] Honda, Michio, Yoshifumi Nishida, Lars Eggert, Pasi Sarolahti, and Hideyuki Tokuda. "Multipath congestion control for shared bottleneck." In Proc. PFLDNeT workshop, vol. 357, p. 378. 2009.

[3] Peng Qiuyu, Anwar Walid, Jaehyun Hwang, and Steven H. Low. "Multipath TCP: Analysis, design, and implementation." IEEE/ACM Transactions on networking 24, no. 1 (2014): 596-609.

[4] Khalili Ramin, Nicolas Gast, Miroslav Popovic, and Jean-Yves Le Boudec. "MPTCP is not Pareto-optimal: Performance issues and a possible solution." IEEE/ACM Transactions On Networking 21, no. 5 (2013): 1651-1665.

[5] F. Kelly, A.K. Maulloo and D. K. Tan. "Rate control for communication networks: shadow prices, proportional fairness and stability." Journal of the Operational Research Society, v49(1), (1998): 237-252.

[6] Kelly Frank, and Thomas Voice. "Stability of end-to-end algorithms for joint routing and rate control." ACM SIGCOMM Computer Communication Review 35, no. 2 (2005): 5-12.

[7] Raiciu Costin, Mark Handley, and Damon Wischik. "Coupled congestion control for multipath transport protocols." (2011).

[8] Constantino M. Lagoa , Hao Che, and Bernardo A. Movsichoff. "Adaptive control algorithms for decentralized optimal traffic engineering in the Internet." IEEE/ACM Transactions on Networking 12, no. 3 (2004): 415-428.

[9] Mehdi Kalantari and Mark Shayman. "Quantifying Responsiveness of TCP Aggregates by Using Direct Sequence Spread Spectrum CDMA and Its Application in Congestion Control.", In Proceedings of IEEE Globecom, 2004.

[10] Wenjing Sun, Chunyu Liu, Constantino Lagoa, Hao Che, Ke Xu, and Yong Cui. "A family of optimal, distributed traffic control laws in a multidomain environment." IEEE Transactions on Control System Technology 23, no. 4 (2015): 1373-1386.

[11] Movsichoff Bernardo A., Constantino M. Lagoa, and Hao Che. "End-to-end optimal algorithms for integrated QoS, traffic engineering, and failure recovery." IEEE/ACM Transactions on networking 15, no. 4 (2007): 813-823.

[12] Lei Ye, Zhijun Wang, Hao Che, and Constantino M. Lagoa. "TERSE: A unified end-to-end traffic control mechanism to enable elastic, delay adaptive, and rate adaptive services." IEEE Journal on Selected Areas in Communications 29, no. 5 (2011): 938-950.

[13] Constantino Lagoa , and Hao Che. "Decentralized optimal traffic engineering in the Internet." ACM SIGCOMM Computer Communication Review v30, no. 5 (2000): 39-47.

[14] Raiciu, Costin, Sebastien Barre, Christopher Pluntke, Adam Greenhalgh, Damon Wischik, and Mark Handley. "Improving datacenter performance and robustness with multipath TCP." ACM SIGCOMM Computer Communication Review 41, no. 4 (2011): 266-277.

[15] Polishchuck Tatiana and Andrei Gurtov. "Improving TCP-friendliness and Fairness for mHIP." Infocommunications Journal, v11, 2011: 26-34.

[16] Andrei Gurtov and Polishchuck Tatiana, "Secure multipath transport for legacy Internet applications." In Proceedings of BROADNETs, 2009.

[17] Steven H. Low, and David E. Lapsley. "Optimization flow control. I. Basic algorithm and convergence." IEEE/ACM Transactions on networking 7, no. 6 (1999): 861-874.

[18] Steven H. Low. "A duality model of TCP and queue management algorithms." IEEE/ACM Transactions On Networking 11, no. 4 (2003): 525-536.

[19] Zhijun Wang, Akshit Singhal, Yunxiang Wu, Chuwen Zhang, Hao Che, Hong Jiang, Bin Liu, and Constantino Lagoa. "HOLNET: A Holistic Traffic Control Framework for Datacenter Networks." In 2020 IEEE 28th International Conference on Network Protocols (ICNP), pp. 1-12. IEEE, 2020.

[20] Lei Ye, Zhijun Wang, Hao Che, Henry BC Chan, and Constantino M. Lagoa. "Utility function of TCP." Computer communications 32, no. 5 (2009): 800-805. Harvard

[21] H. Han, Srinivas Shakkottai, C. V. Hollot, R. Srikant, and D. Towsley. "Overlay TCP for multi-path routing and congestion control." In IMA Workshop on Measurements and Modeling of the Internet. 2004.

[22] Dongsu Han, Robert Grandl, Aditya Akella, and Srinivasan Seshan. "Fcp: a flexible transport framework for accommodating diversity." In Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM, pp. 135-146. 2013.

[23] S. K. Korovin and V. I. Utkin. "Using sliding modes in static optimization and nonlinear programming." Automatica 10, no. 5 (1974): 525-532.

[24] Tabassum Lubna, Imtiaz Mahmud and You-Ze Cho. "D-LIA: Dynamic congestion control algorithm for MPTCP." ICT Express 6, no. 4 (2020): 263-268.

[25] Tabassum Lubna, Imtiaz Mahmud, Geon-Hwan Kim, and You-Ze Cho. "D-OLIA: A Hybrid MPTCP Congestion Control Algorithm with Network Delay Estimation." Sensors 21, no. 17 (2021): 5764.

[26] C. Paasch, S. Barre et al., Multipath TCP in the Linux Kernel, available from http://www.multipath-tcp.org/

[27] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose. "Modeling TCP throughput: A simple model and its empirical validation." In Proceedings of the ACM SIGCOMM'98 conference on Applications, technologies, architectures, and protocols for computer communication, pp. 303-314. 1998.

[28] Morteza Kheirkhah and Wakeman, Ian and Parisis, George. "MMPTCP: A multipath transport protocol for data centers." In IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications, 2016.