Uni-MPTCP($\vec{\omega}, n$): a Unified MPTCP Congestion Control Algorithm

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

***Abstract*—A fundamental design principle of MultiPath TCP (MPTCP) congestion control algorithm (CCA) is that an MPTCP flow should be fair to and do not harm TCP flows. Unfortunately, to deal with cost heterogeneity among subflow interfaces, the existing cost-aware MPTCP CCAs often violate this design principle in an attempt to minimize the cost. Based on the network utility maximization (NUM) framework, we put forward Uni-MPTCP($\vec{\omega}, n$), a NUM-optimal, Unified MPTCP CCA with n subflow paths and a n-dimension weight vector $\vec{\omega}$ with $n-1$ independent elements. Uni-MPTCP($\vec{\omega}, n$) abides by this design principle for arbitrary $\vec{\omega}$ and can be customized to achieve specific cost design objectives with proper adaptation of $\vec{\omega}$. As such, Uni-MPTCP($\vec{\omega}, n$) provides a unified solution to enable cost-aware MPTCP CCAs, while adhering to the design principle. Finally, we put forward an adaptation algorithm for, ω, in Uni-MPTCP($\omega, 2$), aiming at maintaining a target MPTCP flow rate with minimum cost for a cost-heterogeneity case with dual connectivity. The test results based on NS-3 simulation demonstrate that Uni-MPTCP($\omega, 2$) can indeed effectively keep track of a given flow rate target with minimum cost, while adhering to the design principle.**

I. Introduction

By leveraging path diversity widely available in today's Internet, MultiPath TCP (MPTCP) [1], [2] can improve over a traditional single-path TCP in terms of throughput, resource utilization, reliability, and cost saving. In particular, the IETF-standard-based MPTCP congestion control algorithm (CCA), known as Linked Increase Adaptation (LIA) [3], can improve over an Additive-Increase-and-Multiplicative-Decrease (AIMD) based TCP CCA like TCP Reno, while being fair to TCP. LIA was designed based on the following three design principles [4]:

- P1: An MPTCP flow performs at least as well as a TCP flow that uses the best path of all the subflow paths used by the MPTCP flow;
- P2: No subflow in the MPTCP flow occupies more capacity than a TCP flow using the same subflow path.
- P3: MPTCP allows resource pooling, i.e., balancing the subflow rates in the face of congestion.

While the first principle, P1, is considered crucial to incentivize users to use MPTCP, the second one, P2, ensures that MPTCP is fair to and does not harm TCP. The third one, P3, is needed to maximize the MPTCP throughput performance. Subsequent development of variants of LIA (e.g., Balia [5], O-LIA [4] and UMPTCP [6]) and other types of MPTCP CCAs (e.g., C-MPBBR [8] for BBR [9] and mpCubic [10] for TCP Cubic [11]) followed suit, all attempting to adhere to the three design principles, albeit some failed to abide by P1 and/or P2. Note that all but a few early MPTCP CCAs were designed with P3 in mind, including all mentioned here. In what follows, we exclusively focus on P1 and P2, assuming that P3 is automatically satisfied. Moreover, in this paper, we are only concerned with the design of MPTCP CCAs for AIMD-based TCP only.

In this paper, we argue that while P2 is indeed fundamental and should serve as the underlying design principle for all MPTCP CCAs, P1 is not and should be removed for the following two reasons. First, in the face of *cost heterogeneity*, i.e., subflows using different subflow interfaces incur different costs (e.g., in terms of energy [7], [15], [26]–[29], channel quality [16], [17], [20], [21], [30], [31] or monetary cost [20], [21]), P1 may prevent MPTCP from being able to explore the tradeoffs between performance and cost. This is simply because to abide by this principle, an MPTCP flow will have to balance the subflow rates on all the subflow paths in an attempt to perform at least as well as all the TCP flows sharing those subflow paths, regardless of cost. As such, the MPTCP flow often yields to TCP flows on high-load, low-cost subflow paths and heavily rely on low-load, high-cost subflow paths to stay competitive, incurring unnecessary cost. Second, even without cost heterogeneity (i.e., the costs for using different subflow interfaces are the same), P1 may still prevent an MPTCP flow from achieving its full potential, as we shall see in Section III-B.

Unfortunately, the current approach is to either modify or replace an existing MPTCP CCA with a new one to achieve cost-related design objectives (e.g., [7], [15], [20], [21]). By doing so, however, the existing cost-aware MPTCP CCAs not only do away with P1 as expected, but often make it difficult to determine whether or not they still adhere to P2. Moreover, they are directly tailored to specific cost-related design objectives and hence, are point solutions that must be done case by case, separately. For example, different types of costs may call for different cost-related design objectives, e.g., "using lower cost interfaces as much as possible unless the MPTCP flow rate target cannot be achieved" versus "partitioning the flow rate into subflow rates in such a way that the total energy consumption is minimized" [18], [19].

With the above observations, it becomes clear that a more fundamental and systematic solution is warranted. Specifically, a new MPTCP CCA should be designed to replace LIA, so that (a) it abides by P2; (b) it is backward compatible with LIA; and (c) it can be customized to meet additional cost-related design objectives. Such an MPTCP CCA, if successfully designed, provides a unified baseline MPTCP CCA, from which various cost-aware MPTCP CCAs can then be developed, all with provable P2 compliance. The work in this paper aims at achieving this design goal.

The approach taken for the work in this paper is based on the network utility maximization (NUM) framework. We consider TCP flows with a logarithm utility function of the flow rate that captures AIMD behaviors of TCP CCAs like TCP Reno and MPTCP flows with a logarithm utility function of the weighted sum of subflow rates of n subflows, with $n-1$ independent tunable weights expressed in the form of a n-dimension vector $\vec{\omega}$. It makes the following major contributions:

1) We show that the NUM-optimal flow rate allocation for an MPTCP flow co-existing with TCP flows abides by P2 for arbitrary $\vec{\omega}$;
2) We derive Uni-MPTCP($\vec{\omega}, n$), a NUM-optimal unified MPTCP CCA, that achieves the above flow rate allocation and hence, abides by P2. It is also customizable to meet cost-related objectives via proper adaptation of $\vec{\omega}$;
3) We apply Uni-MPTCP($\omega, 2$) to enable an MPTCP flow with two subflow interfaces of different monetary costs (e.g., free WiFi and a pay-as-you-go cellular). By incorporating an adaptation algorithm for, ω, Uni-MPTCP($\omega, 2$) aims at achieving a target MPTCP flow rate with minimum cost;
4) NS-3 simulations show that Uni-MPTCP($\omega, 2$) effectively tracks a target rate with minimal cost under TCP Reno dynamics, while complying with P2.

II. Related Work

MPTCP has gained prominence for improving throughput, reliability, resource utilization, and cost efficiency. Extensive research has been devoted to the design of MPTCP variants to meet these goals.

Early MPTCP CCAs lacked resource pooling (P3), instead treating subflows as independent TCP flows—an approach that proved overly aggressive and unfair to single-path TCP. EWTCP [12] addressed this by reducing each subflow's window growth rate by a factor of $1/n^2$, improving fairness but still lacking P3. Consequently, it limited performance gains in throughput, reliability, and cost.

The MPTCP CCAs developed after EWTCP have taken P3 as an underlying design principle by default, including LIA and its variants (e.g., OLIA [4], Balia [5] and Semi-coupled [22]), and other types of MPTCP CCA that were developed with respect to other TCP CCAs, e.g., mpCubic [10] for TCP Cubic [11] and C-MPBBR [8] for BBR [9]. As mentioned earlier, all these MPTCP CCAs also aimed to fulfill P1 and P2. Note that the reason for the need to use different MPTCP CCAs for different types of TCP CCAs is that TCP CCAs belonging to different types are generally incompatible with one another in terms of aggressiveness in response to network dynamics. For example, the two most widely deployed loss-based TCP CCAs, TCP Reno [13] and TCP Cubic [11], are incompatible with each other. With an AIMD congestion avoidance phase, TCP Reno is much less aggressive than TCP Cubic that grows its congestion window as a cubic function of time [14]. Consequently, to abide by the three design principles, different MPTCP CCAs must be designed for the two TCP CCAs, separately. All the MPTCP CCAs mentioned sofar were designed empirically without considering a global optimization objective, although EWTCP and Semi-coupled turned out to be NUM-optimal, as proven in [6]. More recently, NUM-optimal variants of LIA were designed, including two families of MPTCP CCAs [6], [24] and a hybrid MPTCP CCA [25]. However, all these MPTCP CCAs are cost unaware.

As mobile devices become the dominant means for Internet access, a large number of cost-aware MPTCP CCAs have emerged to address cost-heterogeneity issues, concerning energy efficiency (e.g., [7], [15], [26]–[29]), channel quality (e.g., [20], [21], [30], [31]), and monetary cost [20], [21]. These MPTCP CCAs aim at achieving vastly different cost-related design objectives and hence are point solutions. Moreover, most of them were empirically designed without provable properties, such as optimality and/or compliance with P2.

III. Uni-MPTCP($\vec{\omega}, n$)

A. Preparation

In this section, we introduce the NUM problem, the control laws that solve the NUM problem in general and the TCP utility function.

The NUM Problem: The Network Utility Maximization (NUM) problem can be formally stated as follows [32]:

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

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, $j = 1, 2, ..., m_i$. This formulation is fluid-flow based, meaning that the resource allocated to each flow is measured in flow rate, a real-value variable that can be changed continuously.

NUM-Optimal Control laws: According to [33], with respect to the NUM problem given in Eq. (1), a family of NUM-optimal distributed flow rate control laws for subflow j in flow i is given 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

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

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. The fact that this control law only uses binary information as input for the control means that it is the ideal solution for the development of NUM-optimal, end-to-end congestion control protocols, which only uses source-inferrable binary information, such as timeout or three duplicated ACKs, for the control. The end-to-end TCP, e.g., TCP Reno, is an example of such a protocol.

In summary, given concave utility functions, the associated end-to-end distributed congestion control laws ensure that each flow is regulated independently, allowing the system to converge to a NUM-optimal allocation that maximizes the aggregate utility under dynamic network conditions.

TCP utility function: Although the TCP CCA like TCP Reno was designed empirically, based on the family of control laws presented above, the work in [24] was able to reverse engineer both the TCP slow start phase (SSP) and the (AIMD)-based congestion avoidance phase (CAP) like TCP Reno to derive the corresponding utility functions, implying that TCP Reno is NUM optimal. In this section, we summarize the key results from [24], which will then be used to derive Uni-MPTCP($\vec{\omega}, n$).

Consider the following fluid-flow version of the generic, end-to-end TCP congestion control with both SSP and AIMD-based CAP [23] that captures the main behaviors of TCP Reno.

In the Slow Start Phase (SSP):

$$\dot{x} \;=\; \begin{cases} \alpha x & if \quad cg = 0 \\ -\beta x & if \quad cg = 1, \end{cases} \quad (5)$$

namely, multiplicative increase with coefficient α in the absence of congestion (i.e., cg=0) and multiplicative decrease with coefficient β in the presence of congestion (i.e., cg=1).

In the congestion Avoidance Phase (CAP):

$$\dot{x} \;=\; \begin{cases} \mu & if \quad cg = 0 \\ -\beta x & if \quad cg = 1, \end{cases} \quad (6)$$

namely, additive increase with increasing rate μ in the absence of congestion (i.e., cg=0) and multiplicative decrease with coefficient β in the presence of congestion (i.e., cg=1).

The idea is to reverse engineer the above generic TCP control laws using Eq. (3) to find the corresponding utility function $U_{tcp}(x)$ and $z_{i,j}(t, x, cg)$ for the control laws, if they do exist. The work [23] provides the affirmative answer, which is stated as follows:
In SSP:

$$U_{tcp}(x) = xlog(1 + \frac{\alpha}{\beta}) \quad (7)$$

and

$$z(t, x, cg) = (\alpha + \beta)x. \quad (8)$$

In CAP:

$$U_{tcp}(x) = (\frac{\mu}{\beta} + x)[log(\mu + \beta x) - 1] - x[log(\beta x) - 1] \quad (9)$$

and

$$z(t, x) = \mu + \beta x \quad (10)$$

Assuming $\beta x \gg \mu$, the CAP utility function in Eq. (9) can be approximately written as:

$$U_{tcp}(x_i) \approx \frac{\mu}{\beta} log(x_i), \quad (11)$$

B. Utility Function of MPTCP

With the above preparation, now we are in a position to address the core design challenge, i.e., what MPTCP utility function should be used for MPTCP flows in order to achieve the aforementioned design goal. Once the MPTCP utility function is known, the corresponding MPTCP CCA can then be readily derived from Eq. (3).

To ensure backward compatibility and comparable competitiveness with TCP, the MPTCP utility function should also have two parts, corresponding to the SSP and CAP of TCP, respectively. To this end, we simply reuse the two parts given in Eqs. (7) and (11) with x being replaced by $\sum_{i=1}^{n} \gamma_i y_i$ as the utility of MPTCP, where y_i and γ_i are the subflow rate and subflow weight for the ith subflow of a MPTCP flow. Namely, in SSP:

$$U_{mptcp}(y_1, y_2, ..., y_n) = (\sum_{i=1}^{n} \gamma_i y_i) log(1 + \frac{\alpha}{\beta}) \quad (12)$$

In CAP:

$$U_{mptcp}(y_1, y_2, ..., y_n) \approx \frac{\mu}{\beta} log(\sum_{i=1}^{n} \gamma_i y_i). \quad (13)$$

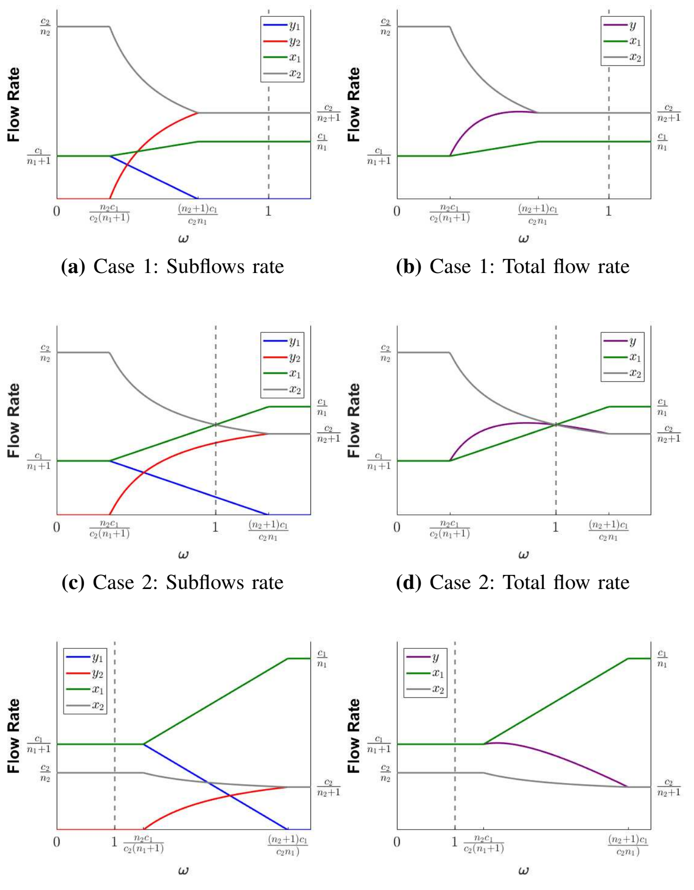
The part in the SSP makes the MPTCP flow as aggressive as TCP Reno at the beginning of the MPTCP session or after a timeout event occurs, which however, is considered rare, relative to the fast recovery events due to the three duplicated acks. In other words, most of the time, both the MPTCP flow and TCP flows are in the CAP, competing for the resources on the subflow paths of the MPTCP flow. This means that to justify the use of this MPTCP utility function, what we need to show is that the NUM-optimal flow rate allocation for the following NUM problem indeed achieves our design goal:

$$Max \; \{\sum_{j=1}^{n} n_j log(x_j) + log(\sum_{i=1}^{n} \omega_i y_i)\} \quad (14)$$

subject to

$$n_i x_i + y_i \le c_i, \qquad for \quad i = 1, ..., n. \quad (15)$$

where y_i, x_i, n_i and c_i are the subflow rate, TCP flow rate, the number of TCP flows and the bottleneck link bandwidth



(a) Case 1: Subflows rate **(b)** Case 1: Total flow rate

(c) Case 2: Subflows rate **(d)** Case 2: Total flow rate

(e) Case 3: Subflows rate **(f)** Case 3: Total flow rate

Fig. 1: Flow rate allocation

on the ith subflow path of the MPTCP flow and $\omega_i = \gamma_i/\gamma_1$ for $i = 1, ..., n$ and $\omega_1 = 1$. Without loss of generality, we assume that $\omega_i \le 1 \ \forall \ i$, given that $\gamma_1 \ge \gamma_i$ for $i = 2, 3, ..., n$. We further denote, $\vec{\omega} = [1, \omega_2, ..., \omega_n]$, i.e., an n-dimension vector with $(n-1)$ independent tunable elements. Note that the MPTCP flow rate $y = \sum_{i=1}^{n} y_i$. For simplicity, in this formulation, we omitted the coefficient $\frac{\mu}{\beta}$ for all logarithmic utility functions and also recast U_{mptcp} in Eq. (13) into the last term in Eq. (14) and omitted the constant term, $log(\gamma_1)$. Obviously both omissions will not affect the NUM-optimal flow rate allocation. Also note that this formulation considers only steady-state flow rate allocation when all the TCP flows sharing the same bottleneck link on any given subflow path i share the same flow rate, x_i.

Now we have the following important results:

Theorem I: The NUM-optimal flow rate allocation for the NUM problem in Eqs. (14) and (15) possesses the following properties: (a) it abides by P2 for arbitrary $\vec{\omega}$; (b) it is backward compatible with the flow rate allocation when LIA is employed to enable the MPTCP flow; and (c) it is inherently customizable to meet additional cost-related design objectives in terms of subflow rate allocation (See appendix A

TABLE I: Flow rate allocation

	$\omega < \frac{n_2 c_1}{(n_1+1)c_2}$	$\frac{n_2 c_1}{(n_1+1)c_2} \le \omega \le \frac{(n_2+1)c_1}{n_1 c_2}$	$\omega > \frac{(n_2+1)c_1}{n_1 c_2}$
x_1	$\frac{c_1}{n_1+1}$	$\frac{c_1+\omega c_2}{n_1+n_2+1}$	$\frac{c_1}{n_1}$
x_2	$\frac{c_2}{n_2}$	$\frac{c_1+\omega c_2}{\omega(n_1+n_2+1)}$	$\frac{c_2}{n_2+1}$
y_1	$\frac{c_1}{n_1+1}$	$c_1 - \frac{n_1(c_1+\omega c_2)}{n_1+n_2+1}$	0
y_2	0	$c_2 - \frac{n_2(c_1+\omega c_2)}{\omega(n_1+n_2+1)}$	$\frac{c_2}{n_2+1}$
y	$\frac{c_1}{n_1+1}$	$c_1 + c_2 - \frac{(n_1\omega+n_2)(c_1+\omega c_2)}{\omega(n_1+n_2+1)}$	$\frac{c_2}{n_2+1}$

for the Proof).

Theorem I means that the NUM-optimal MPTCP CCA corresponding to this logarithm utility with the tunable vector, $\vec{\omega}$, will indeed achieve our design goal.

To further confirm the above claim, we take a close look at an easily solvable case, i.e., the case with $n = 2$. The NUM-optimal flow rate allocation for the case is given in Table I and also plotted in Fig.1 for the three distinct cases with respect to the sole tunable parameter, ω (i.e., ω_2), particularly at $\omega = 1$. For each case, the plot on the left gives the subflow rates, y_1 and y_2, along with x_1 and x_2. Clearly $y_1 \le x_1$ and $y_2 \le x_2$ for all three cases, i.e., abiding by P2, confirming claim (a) in Theorem I. For each case, the plot on the right presents the MPTCP flow rate, y, again along with x_1 and x_2. As one can see, $y \ge x_1$ and x_2 for all cases at $\omega = 1$, i.e., abiding by P1 as well, hence compatible with LIA at $\omega = 1$, confirming claim (b) in Theorem I. Moreover, by inspecting both plots for each case, one should be convinced that by adjusting ω, the relative subflow rates, y_1 and y_2 can indeed be rebalanced, meaning that cost-related objectives can be enforced without violating P2, confirming claim (c) in Theorem I.

Interesting enough, a shrewd reader may have already noticed that y may peak at $\omega \neq 1$, suggesting that abiding by P1, or equivalently, forcing $\omega = 1$, may, in fact, limit the flow rate for an MPTCP flow and hence, should be removed, as claimed earlier. This seemingly counter-intuitive phenomenon can be easily understood by inspecting case I. As ω reduces to be below $\frac{(n_2+1)c_1}{n_1 c_2}$, y_2 starts to drop, yielding to x_2. Meanwhile, y_1 starts to grow from 0, taking bandwidth from x_1, and at certain ω value, y actually peaks. In other words, by allowing TCP flows on some subflow paths to do better and some others to do worse, the MPTCP flow may actually perform better than attempting to do at least equally well as TCP flows on all subflow paths. In fact, from the expression of y in Table I, it can be easily shown that y peaks at $\omega = \sqrt{\frac{c_1 n_2}{c_2 n_1}}$.

Finally, we present useful properties with respect to the three cases that allow them to be uniquely identified by three measurable flow rates:

Corollary I: *The three cases can be uniquely identified by the following criteria:*

- *Case 1: if* $y_0 = y_r > y_l$*,*
- *Case 2: if* $y_0 > y_r$ *and* y_l*,*

- *Case 3: if $\quad y_0 = y_l > y_r$,*

where y_0, y_l, y_r are the Uni-MPTCP($\omega, 2$) flow rates measurable at $\omega = 1$, $\omega \to 0$ and $\omega \to \infty$, respectively.

Proof: We skip the proof as they can be easily confirmed by visual inspection of Fig. 1.

C. Uni-MPTCP($\vec{\omega}, n$)

Similar to the MPTCP utilities, we simply reuse the two parts given in Eqs. (8) and (10) with x being replaced by $\sum_{i=1}^{n} \omega_i y_i$ as the z-functions for MPTCP. Then by applying Eqs. (3) and (4), we arrive at Uni-MPTCP($\vec{\omega}, n$) as follows:

In SSP:

$$\dot{y_l} = \begin{cases} \alpha \omega_l y_l & \text{if } cg_l = 0 \\ -\beta \omega_l y_l & \text{if } cg_l = 1. \end{cases} \tag{16}$$

In CAP:

$$\dot{y_l} \approx \begin{cases} \frac{\omega_l y_l}{\sum_{i=1}^{n} \omega_i y_i} \mu & \text{if } cg_l = 0 \\ -\beta \omega_l y_l & \text{if } cg_l = 1. \end{cases} \tag{17}$$

These control laws are fluid-flow based, and can be easily converted into 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,

$$\alpha \approx 2\beta \approx 1/\tau, \qquad \mu = MSS/\tau \tag{18}$$

Let $y_l = W_l \times MSS/\tau_l$. Then the congestion window size change for subflow l in each RTT, ΔW_l, according to Eqs. (16) and (17), are,

In SSP:

$$\Delta W_l \approx \begin{cases} \omega_l W_l & \text{if } cg_l = 0 \\ -\frac{\omega_l W_l}{2} & \text{if } cg_1 = 1. \end{cases} \tag{19}$$

In CAP:

$$\Delta W_l \approx \begin{cases} \frac{\omega_l W_l}{\tau_l (\sum_{i=1}^{n} \omega_i W_i / \tau_i)} & \text{if } cg_l = 0 \\ -\frac{\omega_l W_l}{2} & \text{if } cg_l = 1. \end{cases} \tag{20}$$

The above window-based Uni-MPTCP($\vec{\omega}, n$) is backward compatible with LIA at $\vec{\omega} = \vec{1}$ and TCP Reno at $n = 1$. Strictly speaking, Uni-MPTCP($\vec{\omega}, n$) is not a protocol, but a family of protocols taking different $\vec{\omega}$ values. For any arbitrary network with any given numbers of active Uni-MPTCP($\vec{\omega}, n$) flows at a given $\vec{\omega}$ and active AIMD-based TCP flows, the network will be guaranteed to converge to a NUM-optimal operational state where the flow rate allocation maximizes the total utility, subject to the link capacity constraints. With an $\vec{\omega}$-adaptation algorithm that properly adjusts $\vec{\omega}$ in response to background TCP flow fluctuations at a timescale much larger than a round-trip-time (RTT), Uni-MPTCP($\vec{\omega}, n$) is expected to be able to keep track of a given cost-related design objective, while being able to converge to a new NUM-optimal operational state in each $\vec{\omega}$-adaption epoch, while adhering P2 and P3.

IV. Application to a Cost-Heterogeneity Case with Dual Connectivity

A. Problem Statement

We consider a cost-heterogeneity case with dual connectivity where the per-byte costs for using two interfaces are different, e.g., a free WiFi interface and a pay-as-you-go 5G interface. The cost-related design objective is to *strive to achieve a target flow rate, λ, with minimum cost.*

An example target flow rate is the video encoding rate for YouTube video streaming, which may be set at a given value by the application or by a user [35]. Note that the target flow rate, λ, may or may not be attainable due to the need for the MPTCP flow to abide by P2. In the context of our network model, We assume that using interface 1 with capacity c_1 costs less than using interface 2 with capacity c_2 (will explain why shortly). A naive solution that provides a guarantee of the target flow rate, λ, is to simply run one subflow as a TCP flow on interface 1, that is, let $y_1 = x_1$ at the bottleneck link and run the other subflow on interface 2 with $y_2 = \lambda - y_1$. Although providing target flow rate guarantee, this solution may violate P2, i.e., it cannot guarantee that $y_2 \leq x_2$. With Uni-MPTCP($\omega, 2$), the compliance with P2 is guaranteed.

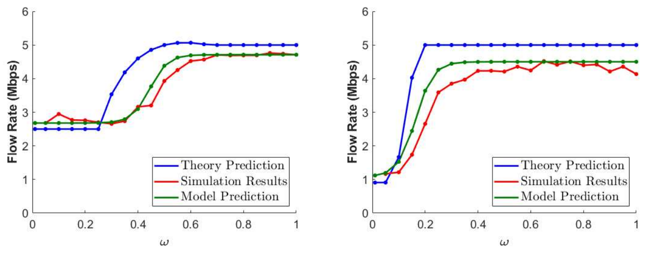
What's left to be done is to design a ω-adaptation algorithm for Uni-MPTCP($\omega, 2$) to achieve the above cost-related design objective. As aforementioned, we assume that using interface 1 costs less than using interface 2. This means that $0 < \omega \equiv \omega_2 \leq 1$, which is desirable for the following reasons. First, note that the flow rate allocation for the MPTCP flow will be in favor of subflow path one when $\omega < 1$. So, to minimize the cost, ω should be as small as possible and should not increase unless the flow rate target, λ, cannot be attained. Furthermore, it should not go beyond one when interface 2 is favored over interface 1, meaning that the cost will become too high to justify the effort to reach λ.

B. A ω-Adaptation Algorithm for Uni-MPTCP($\omega, 2$)

We propose a hybrid measurement-and-modeling-based adaptation algorithm for ω in $(0, 1]$ to achieve the cost-related design objective.

The idea is to only measure the flow rates, y_0, y_l and y_r at $\omega = 1, 0.01$ and 100, respectively (note that in practice, sampling ω at 0.01 and 100 is the same as sampling at 0 and ∞). The three measured flow rates are then used to identify which of the three cases it is, based on the criteria given in Corollary I. If it is Case 3, set $\omega = 0.01$, as y does not change with ω and the search is done. Otherwise use the following logistic function with four parameters, α, β, γ and k, to model the flow rate curve for both Cases 1 and 2 in $0 < \omega \leq 1$,

$$y = \frac{\gamma}{1 + e^{-k(\omega - \beta)}} + \alpha. \tag{21}$$



(a) $n_1 = 3, n_2 = 1$ **(b)** $n_1 = 10, n_2 = 1$

Fig. 2: Test of hybrid model

This function is found to be able to fit the simulated flow rate curves well, when k is in double digits, e.g., $k = 30$ and β is the ω value corresponding to the flow rate y at about one-third of the highest achievable flow rate, i.e., y at $\omega = 1$ for this logistic function. With this understanding, we can then estimate α and γ as follows:

$$\alpha \approx y_l, \tag{22}$$

and

$$\alpha + \gamma \approx y_r, \tag{23}$$

or

$$\gamma \approx y_r - y_l, \tag{24}$$

assuming that the exponential term in the denominator approaches 0 and ∞, when $\omega \to \infty$ and 0, respectively.

Finally, with the combination of both a series of approximations, and trial and error, we find that,

$$\beta = \frac{\sqrt{\frac{y_l}{y_r}} + 2 \times max\{2 \times \frac{y_l}{y_r} - \sqrt{\frac{y_l}{y_r}}, 0\}}{3} \tag{25}$$

matches the ω value at the one-third of the highest achievable flow rate well.

The hybrid MPTCP flow rate prediction model is evaluated through simulations conducted in the open-source NS-3 network simulator [36], which has been extended to support MPTCP. We consider a low-bandwidth-and-low-delay scenario where TCP Reno and the associated MPTCP like Uni-MPTCP($\vec{\omega}, n$) and LIA work well (note that for high-bandwidth-and-high-delay networks, TCP Cubic and the associated MPTCP work better). Namely, we set $c_1 = c_2 = 10$ Mbps and the end-to-end propagation delay at 0.4 ms for all flows. All the TCP flows run TCP Reno based on the simulation code provided by NS-3. For each given n_1 and n_2 pair, 10 simulation runs of 300 seconds each are performed at a given ω, from which the average Uni-MPTCP($\omega, 2$) flow rate is recorded. This is repeated for ω taking values from 0.05 to 1 with step length of 0.05.

Fig. 2 gives the flow rate curves based on theory (Table I), hybrid model and measurement. Note that due to the use of a fluid flow model versus a packet-based model, the gap between the theory and measurement is inevitable. On one hand, the theory predicts $y_2 = 0$ as $\omega \to 0$, whereas for Uni-MPTCP($\omega, 2$), the smallest y_2 can reach is one maximum segment size (MSS) per RTT, higher than that predicted by the theory. On the other hand, Uni-MPTCP($\omega, 2$) always achieves a lower high flow rate, y, than that predicted by the theory, due to the congestion feedback delays and discrete flow rate adaptation. In contrast, the hybrid model matches the measurement much better.

Finally, We note that the peak of y predicted by the theory in the left plot is barely visible by the measurement. This is because at $n = 2$, it can be shown (not given in the paper) that the achievable peak relative to y_0 at $\omega = 1$ is upper bounded at 8.7%. For this reason, our hybrid model simply assumes that y is a non-decreasing function of ω in $[0, 1]$ without attempting to capture the peak.

With the above hybrid model, the ω-adaptation algorithm for each search round is presented in Algorithm I. The algorithm may be run periodically at a given interval T or on-demand (e.g., triggered by measured changes of y_1 and y_2) in response to background TCP traffic fluctuations.

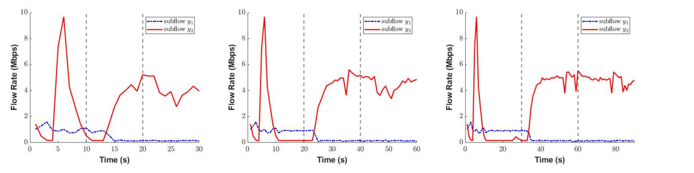
Algorithm 1 ω-adaptation algorithm for cost-aware Uni-MPTCP($\omega, 2$)

```
Input:
  $target$ : Target flow rate
Output: a $\omega$ value
Begin
  // Run Uni-MPTCP($\omega, 2$) at $\omega = 0.01, 100, 1$ with 20
  seconds each, to measure $y_l$, $y_r$, $y_0$, respectively
if $Case3$ then
    Return $\omega$ = 0.01
else
    // Estimate $\alpha$, $\beta$ and $\gamma$ for logistic function
    // Calculate the peak flow rate from logistic function
end if
if $target < y_l$ then
    Return $\omega$ = 0.01
else if $target >$ Peak flow rate then
    Return $\omega$ = 1
else
    // calculate the $\omega$ value corresponding to Target from
  logistic function
    Return $\omega$ value
end if
End
```

C. Test of Uni-MPTCP($\omega, 2$)

In this section, we test cost-aware Uni-MPTCP($\omega, 2$) under both stable and dynamically changing network conditions based on NS-3 simulation. We consider the same scenario studied in the previous section.

1) Per-sample Search Time: First, we need to test and see what per-sample search time (i.e., the ω sampling epoch) and wait or measurement period should be used. Consider $n_1 = 10$, $n_2 = 1$, i.e., case 1, and three different lengths of per-sample search time: 10, 20 and 30 seconds. Fig. 3 depicts the measured Uni-MPTCP($\omega, 2$) flow rate over the entire search round that involves three lengths of per-sample search time, at $\omega = 0.01$,



(a) $10s$ **(b)** $20s$ **(c)** $30s$

Fig. 3: Different lengths of per-sample search time

TABLE II: Performances of different sets of per-sample search time

	y_l (Mbps)	y_r (Mbps)	y_0 (Mbps)	ω decision
Long run result	1.120801	4.50132	4.50132	0.152217
Measure 0s-10s	2.72832	3.24272	3.463455	0.01
Measure 0s-20s	2.05799	4.29819	4.0166	0.360981
Measure 0s-30s	1.77797	4.0276	4.3877	0.342145
Measure 5s-10s	1.51796	3.18734	3.5483	0.416855
Measure 5s-20s	1.30549	4.56857	4.08284	0.184515
Measure 5s-30s	1.23795	4.13308	4.36794	0.208344
Measure 10s-20s	1.16627	4.67518	4.37245	0.150182
Measure 10s-30s	1.1489	4.18582	4.45983	0.183954
Measure 15s-20s	1.20071	4.60178	4.40733	0.161596
Measure 15s-30s	1.15639	4.07374	4.41856	0.19562

100 and 1, in that order. First, the big spike found in the first sampling epoch for all three cases is caused by the initial SSP. This makes the 10-second case in Fig. 3(a) unable to stabilize throughout the entire epoch. In contrast, for both the 20-second and 30-second cases in Fig. 3 (b) and (c), the flow rates stabilize before the end of the first half of the epoch, which appears to be also true for the other two epochs. In other words, while for the 10-second case it is hard to find a measurement period, at least for the first epoch, in which reliable NUM-optimal flow rate can be measured, for both of the other two cases, letting the measurement period to be the second half of each epoch for all three epochs appears to work well. Furthermore, as aforementioned, one should keep the search time small to make Uni-MPTCP($\omega, 2$) as responsive to fluctuation of background traffic as possible. For this reason, we would prefer the 20-second case over the 30-second one.

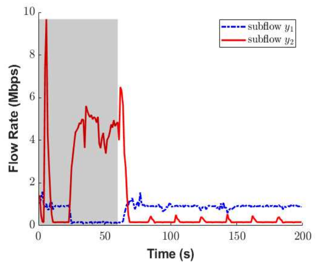
2) Steady-State Case Study: In this case study, we assume that the background traffic load is stable, meaning that throughout the experiment, the numbers of TCP flows on the two subflow bottleneck links are fixed at $n_1 = 10$ and $n_2 = 1$, i.e., Case 1. Furthermore, we consider the following cost model:

$$C_{total} = P_{c1} \times y_1 + P_{c2} \times y_2 \tag{26}$$

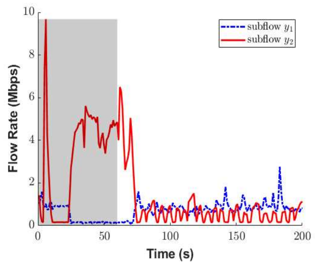
where C_{total} is the total Uni-MPTCP($\omega, 2$) flow cost per unit time, and P_{c1} and P_{c2} are the per-byte costs for using the low-cost and high-cost links, respectively, and we set, $P_{c1} = 0.1$ and $P_{c2} = 10$.

We consider 5 different cases where the Uni-MPTCP($\omega, 2$) flow strives to attain 5 different target flow rates: 1.0 Mbps, 2.5 Mbps, 3.8 Mbps, 5.0 Mbps and 10 Mbps. These cases are selected to cover three different possible scenarios. First, the 1.0 Mbps target can be fulfilled almost entirely by the low-cost link with ω taking a small value. Second, both the 2.5 Mbps and 3.8 Mbps targets fall in the region in which the ω-adaptation algorithm will be fully activated, called effective region hereafter. Third, both the 5.0 Mbps and 10.0 Mbps targets are higher than the highest achievable flow rate and hence, cannot be attained. In this case, the flow rate should converge to the highest achievable rate.

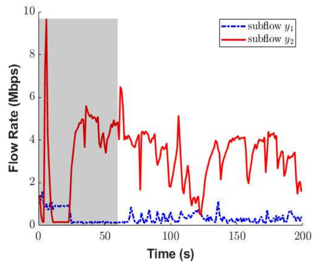
The results are presented in Table III. It also includes the result by running LIA. The purpose is to see how much one can benefit from the cost-aware Uni-MPTCP($\omega, 2$) in terms of cost savings, compared to a cost-unaware protocol like LIA. First of all, we note that Uni-MPTCP($\omega, 2$) attains a flow rate lower than the corresponding target, except for the case of 1 Mbps target. This can be understood with the reasoning given in the previous section. Second, we note that Uni-MPTCP($\omega, 2$) tends to give relatively higher prediction errors when the targets fall in the effective region. This is because in this region, Uni-MPTCP($\omega, 2$) needs to balance the loads between the two



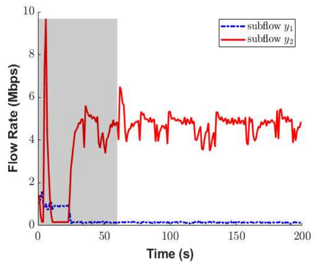
(a) Target rate = 1.0Mbps



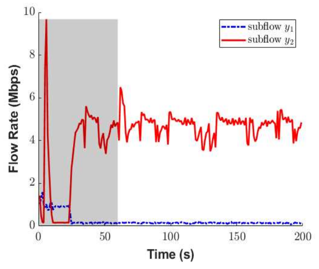
(b) Target rate = 2.5Mbps



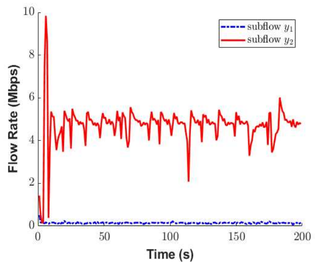
(c) Target rate = 3.8Mbps



(d) Target rate = 5.0Mbps



(e) Target rate = 10.0Mbps



(f) LIA

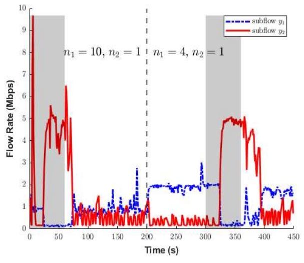
Fig. 4: Different user required data rate

TABLE III: Uni-MPTCP($\omega, 2$) flow rates and costs at different target flow rates

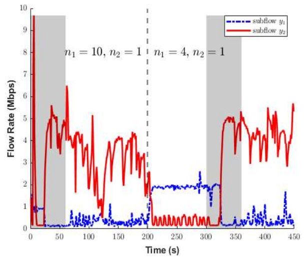
	y (Mbps)	y_1 (Mbps)	y_2 (Mbps)	C_{total}
LIA	4.8715	0.1425	4.729	47.30425
Uni-MPTCP($\omega, 2$)(1.0Mbps)	1.0982	0.9085	0.1897	1.98785
Uni-MPTCP($\omega, 2$)(2.5Mbps)	1.79725	0.8425	0.95475	9.63175
Uni-MPTCP($\omega, 2$)(3.8Mbps)	3.5173	0.342	3.1753	31.7872
Uni-MPTCP($\omega, 2$)(5.0Mbps)	4.89518	0.1438	4.75138	47.52818
Uni-MPTCP($\omega, 2$)(10.0Mbps)	4.89518	0.1438	4.75138	47.52818

TABLE IV: Uni-MPTCP($\omega, 2$) flow rates in distinct time window

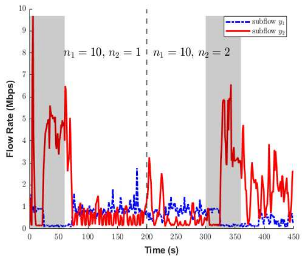
	[60s-200s]			[200s-300s]			360s onward		
	y	y_1	y_2	y	y_1	y_2	y	y_1	y_2
Scenario 1: Target rate=2.5Mbps	1.79725	0.8425	0.95475	2.2373	1.8929	0.3444	2.343	1.6785	0.6645
Scenario 1: Target rate=3.8Mbps	3.5173	0.342	3.1753	2.3086	1.8487	0.4599	4.1822	0.2991	3.8831
Scenario 2: Target rate=2.5Mbps	1.79725	0.8425	0.95475	1.4184	0.7766	0.6418	2.2716	0.365	1.9066
Scenario 2: Target rate=3.8Mbps	3.5173	0.342	3.1753	1.446	0.6553	0.7907	3.5667	0.1519	3.4148



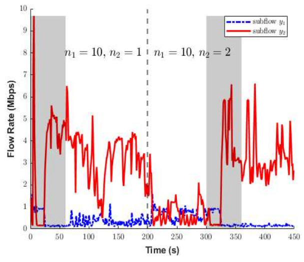
(a) Scenario 1: Target rate=2.5Mbps



(b) Scenario 1: Target rate=3.8Mbps



(c) Scenario 2: Target rate=2.5Mbps



(d) Scenario 2: Target rate=3.8Mbps

Fig. 5: Dynamic network environment case study

subflow paths, resulting in higher subflow rate fluctuations as we shall see shortly. Moreover, although much improved over the theoretical one, the hybrid solution that attempts to capture the flow rate changes in the effective region can still contribute significantly to the flow rate prediction errors. This explains why the 2.5 Mbps target is not very well captured. Third, for the two targets above the highest achievable flow rate, Uni-MPTCP($\omega, 2$) gives the same flow rate allocation at $\omega = 1$, which is slightly higher than that of LIA with slightly higher cost, as expected. Finally, it becomes clear that Uni-MPTCP($\omega, 2$) can provide significant cost savings over a cost-oblivious protocol, such as LIA.

We also show Uni-MPTCP($\omega, 2$) and LIA in action in Fig. 4. Only one search round is performed at the beginning (i.e., the shaded area in each plot). It confirms that Fig. 4 (b) and (c), corresponding to the cases where the targets fall in the effective region, are the most unstable ones due to subflow load balancing. It also confirms that LIA is indeed neither cost, nor target flow rate aware.

3) Dynamic Case Study: We consider two scenarios, both with abrupt TCP flow load changes at 200 seconds. In scenario 1, $n_1 = 10$ and $n_2 = 1$ before 200 seconds and $n_1 = 4$ and $n_2 = 1$ afterwards. In scenario 2, again, $n_1 = 10$ and $n_2 = 1$ before 200 seconds and $n_1 = 10$ and $n_2 = 2$ afterwards. For all the TCP load cases of the two scenarios, both the 2.5 Mbps and 3.8 Mbps target flow rates fall in the effective region. In other words, they are hard to keep track of, as the two subflow rates need to be re-balanced after the load change for both scenarios.

We further assume that the searches are done periodically with internal $T = 300$ seconds. This means that after the flow load change at 200 seconds, there is a 100-second time window in which the Uni-MPTCP($\omega, 2$) flow rate may drift away from the target flow rate. Consequently, we have three distinct time windows:

- $[60s, 200s]$: after the first search round and before the TCP load change;
- $[200s, 300s]$: after the TCP load change and before the second search round;
- $360s$ onward: After the second search round.

The average Uni-MPTCP($\omega, 2$) flow rates in these time windows for both scenarios are given in Table IV. First, before TCP load changes, both scenarios give the same flow rate allocation that tracks the corresponding target flow rates, as the TCP loads are the same. Second, in the second time window, the allocated flow rates indeed drift away from their respective targets. Third, after the second search round, the flow rates are able to track their respective targets again, as expected. Fig. 5 also depicts the target tracking process in action for the two scenarios.

Finally, we note that a possible implementation of Uni-MPTCP($\omega, 2$) is to implement Uni-MPTCP($\omega, 2$) in the Kernel with a user-interface API that allows a ω value in Uni-MPTCP($\omega, 2$) to be passed from the user space to the kernel. This makes it possible to design ω-adaptation algorithms to achieve a wide range of design objectives in the user space.

V. Conclusions

In this paper, on the basis of the NUM framework, we first argue that the first of the three MPTCP design principles should be abandoned. Then we put forward Uni-MPTCP($\vec{\omega}, n$), a NUM-optimal multipath congestion control protocol for MPTCP flows with n subflow paths that abides by the remaining two MPTCP design principles at arbitrary $\vec{\omega}$, an n-dimension vector with $(n-1)$ independent elements, which can be adapted to realize any specific cost-related design objectives. Finally, with the design of an adaptation algorithm for ω for a cost-heterogeneity case with dual connectivity, we demonstrate that Uni-MPTCP($\omega, 2$) can effectively keep track of a given multipath flow rate target with minimum cost.

Acknowledgment

This work was supported by the U.S. National Science Foundation under grants CCF-SHF-2008835 and CCF-SHF-2226117.

References

[1] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar, "Architectural Guidelines for Multipath TCP Development," RFC 6182, RFC Editor, Mar. 2011.

[2] O. Bonaventure, C. Paasch, and F. Duchene, "TCP Extensions for Multipath Operation with Multiple Addresses," RFC 8684, Mar. 2020. [Online]. Available: https://www.rfc-editor.org/info/rfc8684

[3] C. Raiciu, M. J. Handley, and D. Wischik, "Coupled Congestion Control for Multipath Transport Protocols," RFC 6356, RFC Editor, Oct. 2011.

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

[5] Q. Peng, A. Walid, J. Hwang, and S. H. Low, "Multipath TCP: Analysis, design, and implementation," *IEEE/ACM Transactions on Networking*, vol. 24, no. 1, pp. 596–609, 2014.

[6] A. Singhal, X. Wang, Z. Wang, H. Che, and H. Jiang, "Two Families of Optimal Multipath Congestion Control Protocols," in *Proceedings of the 2022 IEEE 30th International Conference on Network Protocols (ICNP)*, 2022.

[7] J. Zhao, J. Liu, H. Wang, C. Xu, and H. Zhang, "Multipath congestion control: Measurement, analysis, and optimization from the energy perspective," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 6, pp. 3295–3307, 2023.

[8] I. Mahmud, T. Lubna, Y.-J. Song, and Y.-Z. Cho, "Coupled Multipath BBR (C-MPBBR): An Efficient Congestion Control Algorithm for Multipath TCP," *IEEE Access*, vol. 8, pp. 165497–165511, 2020.

[9] N. Cardwell, Y. Cheng, C. S. Gunn, and S. H. Yeganeh, "BBR: Congestion-Based Congestion Control: Measuring Bottleneck Bandwidth and Round-trip Propagation Time," *ACM Queue*, vol. 14, no. 5, pp. 20–53, 2016.

[10] T. Kato, S. Haruyama, R. Yamamoto, and S. Ohzahata, "mpCUBIC: A CUBIC-like Congestion Control Algorithm for Multipath TCP," in *Trends and Innovations in Information Systems and Technologies: Volume 2*, Springer, 2020.

[11] S. Ha, I. Rhee, and L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant," *ACM SIGOPS Operating Systems Review*, vol. 42, no. 5, pp. 64–74, 2008.

[12] M. Honda, Y. Nishida, L. Eggert, P. Sarolahti, and H. Tokuda, "Multipath congestion control for shared bottleneck," in *Proceedings of the PFLDNeT Workshop*, vol. 357, pp. 378, 2009.

[13] V. Jacobson, "Congestion Avoidance and Control," in *Proceedings of the ACM SIGCOMM Computer Communication Review*, vol. 18, no. 4, pp. 314–329, 1988.

[14] H. Jamal and K. Sultan, "Performance analysis of TCP congestion control algorithms," *International Journal of Computers and Applications*, 2008.

[15] C. Pluntke, L. Eggert, and N. Kiukkonen, "Saving mobile device energy with multipath TCP," in *Proc. 6th Int. Workshop MobiArch*, 2011.

[16] X. Corbillon, R. Aparicio-Pardo, and N. Kuhn, "Cross-layer Scheduler for Video Streaming over MPTCP," in *Proceedings of the 7th ACM International Conference on Multimedia Systems (MMSys)*, 2016.

[17] N. Al-Imareen and G. Lencse, "Real-Time Video Streaming in MPT-GRE Multipath Networks," in *Proceedings of IEEE International Conference on SoftCOM*, 2024.

[18] M. J. Shamani, "Factors Impacting Performance of Multipath TCP," *UNSW Repository*, 2018.

[19] J. Zhao, J. Liu, H. Wang, C. Xu, and W. Gong, "Measurement, Analysis, and Enhancement of Multipath TCP Energy Efficiency for Datacenters," *IEEE/ACM Transactions on Networking*, vol. 27, no. 5, pp. 1925–1938, 2019.

[20] M. Pieska, A. Rabitsch, A. Brunstrom, A. Kassler, M. Amend, and E. Bogenfeld, "Low-delay cost-aware multipath scheduling over dynamic links for access traffic steering, switching, and splitting," *Comput. Netw.*, vol. 241, Art. no. 110186, 2024.

[21] Cisco Technology Inc., "Traffic distribution approaches in multipath TCP with monetary link-cost awareness," U.S.PatentNo.10951514B2, Mar.16,2021.

[22] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath TCP," in *Proc. 8th USENIX Symp. Networked Syst. Design and Implementation (NSDI)*, 2011.

[23] L. Ye, Z. Wang, H. Che, and C. 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*, vol. 29, no. 5, pp. 938–950, 2011.

[24] Z. Wang, A. Singhal, Y. Wu, C. Zhang, H. Che, H. Jiang, B. Liu, and C. Lagoa, "Holnet: A holistic traffic control framework for datacenter networks," in *Proc. IEEE 28th International Conference on Network Protocols (ICNP)*, 2020.

[25] A. Singhal, X. Wang, Z. Wang, H. Che, and H. Jiang, "Hybrid multipath congestion control," *Int. J. Comput. Inf. Eng.*, vol. 16, no. 11, pp. 549–554, 2022.

[26] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: A measurement study and implications for network applications," in *Proc. 9th ACM SIGCOMM Conference on Internet Measurement (IMC)*, 2009.

[27] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A close examination of performance and power characteristics of 4G LTE networks," in *Proc. 10th International Conference Mobile System, Application, and Services (MobiSys)*, 2012.

[28] Y.-S. Lim, Y.-C. Chen, E. M. Nahum, D. Towsley, R. J. Gibbens, and E. Cecchet, "Design, implementation, and evaluation of energy-aware multi-path TCP," in *Proc. 11th ACM Conference Emerging Networking Experiments Technologies (CoNEXT)*, 2015.

[29] L. Zhong, X. Ji, Z. Wang, J. Qin, and G.-M. Muntean, "A Q-learning driven energy-aware multipath transmission solution for 5G media services," *IEEE Trans. Broadcast.*, vol. 68, no. 2, pp. 559–571, 2022.

[30] A. Elgabli and V. Aggarwal, "SmartStreamer: Preference-aware multipath video streaming over MPTCP," *IEEE Trans. Veh. Technol.*, vol. 68, no. 7, pp. 6975–6984, 2019.

[31] P. Hurtig, K.-J. Grinnemo, A. Brunstrom, S. Ferlin, O. Alay, and N. Kuhn, "Low-latency scheduling in MPTCP," *IEEE/ACM Transactions on Networks*, vol. 27, no. 1, pp. 302–315, 2019.

[32] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: Shadow prices, proportional fairness and stability," *J. Oper. Res. Soc.*, vol. 49, no. 3, pp. 237–252, 1998.

[33] W. Su, C. Liu, C. Lagoa, H. Che, K. Xu, and Y. Cui, "A family of optimal, distributed traffic control laws in a multidomain environment," *IEEE Trans. Control Syst. Technol.*, vol. 23, no. 4, pp. 1373–1386, 2015.

[34] G. D. Birkhoff, "Proof of the Arithmetic Mean-Geometric Mean Inequality," *Annals of Mathematics*, vol. 9, no. 3, pp. 296–300, 1908.

[35] Ł. P. Łuczak, P. Ignaciuk, and M. Morawski, "Evaluating MPTCP Congestion Control Algorithms: Implications for Streaming in Open Internet," *Future Internet*, vol. 15, no. 10, p. 328, 2023.

[36] M. Kheirkhah, I. Wakeman, and G. Parisis, "MMPTCP: A multipath transport protocol for data centers," in *Proc. IEEE INFOCOM*, San Francisco, CA, USA, Apr. 2016, pp. 1–9.

Appendix A
Proof of Theorem I

First, with the property of logarithmic functions, Eq. (14) is equivalent to:

$$Max\ \{(\sum_{i=1}^{n}\omega_i y_i)\prod_{i=1}^{n}x_i^{n_i}\}, \tag{27}$$

subject to,

$$n_i x_i + y_i \le c_i, \qquad for \quad i = 1, ..., n. \tag{28}$$

First, we consider the case where $\vec{\omega} = \vec{1}$, i.e., $\omega_i = 1\ \forall\ i$, and by applying the AM-GM inequality [34], we have,

$$y\prod_{i=1}^{n}x_i^{n_i} \le (\frac{y + \sum_{i=1}^{n} n_i x_i}{N})^N \equiv (\frac{\sum_{i=1}^{n} c_i}{N})^N, \tag{29}$$

where $N = n + \sum_{i=1}^{n} n_i$, is the total number of TCP flows and subflows of the MPTCP flow. The equality holds if and only if $y = x_1 = x_2 = ... = x_n$ [34]. The last equality always holds true, assuming that the TCP flows and subflow on each subflow path can always saturate the bottleneck link bandwidth of the subflow path.

The above result simply states that the NUM-optimal flow rate allocation is $y = x_1 = x_2 = ... = x_n$, when the product on the left of the inequality is maximized. This is true as long as the MPTCP subflows can be balanced to make this happen, even subject to the subflow bottleneck link constraints in Eq. (15). Now consider the case where some subflow bottleneck link, say, the kth link, is the highest loaded one, and even with $y_k = 0$, x_k $(= c_k/n_k)$ is still too small to be balanced with y and other x_i's. In this case, the MPTCP flow will have $y_k = 0$ and attempt to balance the rest of the subflows so that $y = x_1 = x_2 = ... = x_n$, except for x_k $(= c_k/n_k)$ to maximize the left of the inequality. The process will repeat, if the next highest loaded link again cannot be balanced, until the MPTCP flow can balance the loads for the remaining lower loaded links with larger x_i's. This proves that for the NUM-optimal flow rate allocation, $y = max\{x_1, x_2, ..., x_n\}$, meaning that at $\vec{\omega} = \vec{1}$, the NUM-optimal flow rate allocation abides by P1. Furthermore, the subflow rate, y_i, cannot be larger than x_i, $\forall$ i, because otherwise, by applying the AM-GM inequality to $y_i \prod_{i=1}^{n_i} x_i$, which is part of the term on the left of the inequality, can be increased by reducing y_i, given that $y_i + n_i x_i \equiv c_i$ is a constant. This means that at $\vec{\omega} = \vec{1}$, the NUM-optimal flow rate allocation also abides by P2. In summary, the NUM-optimal MPTCP CCA at $\vec{\omega} = \vec{1}$ is compatible with LIA, which proves that (b) is true.

Second, for the case where $\vec{\omega} \neq \vec{1}$, i.e., at least one element in $\vec{\omega}$ is not equal 1. Consider $\omega_i < 1$ $(i > 1)$. It is clear that at $\omega_i = 0$, $y_i = 0$, and y_i increases with ω_i until it reaches its maximum at $\omega_i = 1$. Based on the same argument above, this maximum value of y_i cannot exceed x_i. Hence, for any given $\vec{\omega}$, the flow rate allocation abides by P2, which proves that (a) holds true. Furthermore, since changing ω_i changes y_i, by properly adjusting the relative values among ω_i's, one can balance subflow rates to achieve specific cost-related design objectives, which proves that (c) also holds true.