wCompound: Enhancing Performance of Multipath Transmission in High-speed and Long Distance Networks

Rui Zhuang[1], Yitao Xing[1], Wenjia Wei[2], Yuan Zhang[2], Jiayu Yang[1], Kaiping Xue[1,2,*]
[1] School of Cyber Security, University of Science and Technology of China, Hefei, Anhui 230027 China
[2] Department of EEIS, University of Science and Technology of China, Hefei, Anhui 230027 China
[*] Corresponding author, kpxue@ustc.edu.cn

***Abstract*—As the user demand for data transmission over high-speed and long distance (hereafter abbreviated as HSLD) networks increases significantly, multipath TCP (MPTCP) shows a great potential to further improve the utilization of HSLD network resources than traditional TCP, and provides better quality of service (QoS). It has been reported that TCP causes serious waste of bandwidth in HSLD networks, while MPTCP can transmit data by using multiple network paths simultaneously between two distant hosts, thus provides better resource utilization, higher throughput and smoother failure recovery for applications. However, the existing multipath congestion control algorithms cannot perfectly meet the efficiency requirements of HSLD network, since they mainly emphasize fairness rather than other critical indicators of QoS such as throughput, but still encounter fairness issues when coexist with various TCP variants. To solve these problems, we develop weighted Compound (wCompound), a loss-and-delay-based compound multipath congestion control algorithm which is originated from Compound TCP, and is applicable to HSLD networks. Different from the traditional methods of setting an empirical value as the threshold, wCompound innovatively adopts a dynamic threshold and have the flexibility to adjust the sending window of each subflow based on current network state, so as to effectively couple all subflows and fully utilize the network capacity. Moreover, with the cooperation of delay-based and loss-based methods, wCompound also ensures good fairness to different types of TCP variants. We implement wCompound in the Linux kernel, then carry out sufficient experiments on our testbed. The results show that wCompound achieves higher utilization of network resources and can always maintain an appropriate throughput no matter competing with loss-based or delay-based network traffic.**

***Index Terms*—Multipath TCP, coupled congestion control, loss-and-delay-based, high-speed and long distance network**

I. Introduction

Nowadays, high-speed and long distance (HSLD) network is widely used across the world [1] and is becoming increasingly commonplace in modern networks, such network usually has a high bandwidth delay product (BDP). Currently, next-generation networks are developing towards high speed and long delay, reflected in two aspects. Firstly, due to the vigorous growth of Internet applications such as online video, online games and life services, users' demand for bandwidth is increasing sharply. Secondly, the increasing demand for Internet applications also results in the enlargement of data amount and globally expansion of data centers. The traffic generated by data centers will account for most of the total traffic around the world. Among them, traffic transmitted between data centers is likely to establish cross-boundary, or even cross-continent connections, and such connections are quite possible to face the situation of high bandwidth and long delay. Therefore, to fulfill the demand of customers for network bandwidth, so as to ensure good quality of service and user experience, as well as to improve the connectivity between data centers and take great advantage of network resources, the transmission optimization over HSLD networks is a valuable research topic.

Unfortunately, HSLD network is not a typical environment for which the standard TCP is designed. TCP has been shown to substantially underutilize network bandwidth over HSLD networks by some research, and the QoS provided to applications is also far from satisfaction [2]. The existing TCP congestion control algorithms can be mainly classified into four types, namely delay-based, loss-based, compound (Compound TCP [2]) and congestion-based (BBR [3]), but most of them have poor performance in HSLD scenario [2]. Specifically, loss-based approaches are highly aggressive to improve the utilization efficiency of high-speed network links, which cause severe RTT unfairness and TCP unfairness. While delay-based approaches have problems coexisting with loss-based flows, and loss-based flows are likely to occupy delay-based flows' bandwith than their fair share. And for BBR, a novel method for solving the bufferbloat problem in HSLD network, its mechanism inherently causes a massive amount of packet retransmission, and also suffers from serious performance imbalance and excessive packet loss while competing with loss-based flows on a shared link [4] (especially when BBR shares the same link with CUBIC [5]). On the other hand, although there are scholars believe that the transition of congestion control algorithms from loss-based to delay-based is inevitable [6], this transition cannot be completed overnight. Loss-based congestion controls are still widely deployed and many other strategies still coexist in current networks. So a transitional approach is needed, which should be able to coexist and perform well with the varied existing TCP variants. In response to these problems, Compound TCP (CTCP) provides a very good instance, since it combines loss-based and delay-based methods, which can complement each other in practice. CTCP introduces an extra scalable delay-based component in standard TCP, and can maintain good fairness to regular TCP flows while obtaining free network resources more efficiently, which meets all the requirements of high-speed protocol [2]. All in all, CTCP are of great reference value to solve the problems of the existing algorithms in HSLD networks.

MPTCP is an extension of TCP, and is of great benefit to the efficiency improvement of HSLD network. We list the QoS benefits induced by the use of MPTCP: Firstly, MPTCP can transmit data through multiple network interfaces / paths concurrently, which can obtain aggregate bandwidth and increase the throughput, thus it can utilize network resources more efficiently than traditional TCP. Secondly, benefits from the use of multiple paths, MPTCP can provide better robustness and smoother failure recovery. Finally, since MPTCP can shift load between paths, it can effectively reduce packet loss by avoiding sending traffic to routers with full buffers, thus avoid serious delay in the overall transmission. Therefore, compared to TCP, MPTCP has greater potential for performance improvement in HSLD networks.

However, in HSLD networks, the existing various MPTCP schemes do not give full play to the advantages of MPTCP (see Section II). As stated earlier, the single-path scheme CTCP performs well in HSLD networks, and its advantages will be further magnified if it is reasonably extended to a multipath scheme. Therefore, we expect to extend CTCP to MPTCP with the aim of handling multipath transmission in HSLD network efficiently and addressing the problems in the existing algorithms. But we do not simply and directly introduce CTCP into MPTCP. Instead, we aim at coupling all the subflows together so as to achieve both effciency and fairness.

The main contributions of this paper are as follows:

- We propose an improved MPTCP congestion control algorithm adapted to HSLD network, namely wCompound. To effectively couple all the subflows, we design the utility function of wCompound and derive the weights for each subflow based on Network Utility Maximization (NUM), and adaptively adjust the weights to ensure that all flows can share the network resources fairly and efficiently. Different from traditional methods, wCompound uses an adaptive threshold rather than configuring a constant threshold to determine when to back off or forge ahead. Therefore, wCompound flows can obtain more accurate and flexible congestion control. A fine-grained load balancing is realized through the accurate adjustment of weights, so the traffic can be effectively transferred from congested paths to noncongested paths. Consequently, wCompound can utilize the free available bandwidth more sufficiently, and since wCompound will not send too many packets on congested paths, the packet loss caused by buffer overflow is greatly reduced, which avoids serious increase of delay.
- A key challenge of wCompound is to derive an appropriate weight for each subflow to achieve reasonable bandwidth allocation of the network. To this end, we formulate the network bandwidth allocation problem as a NUM problem, and transform it into a solvable problem by using an approximate iterative algorithm. Through theory analysis, we obtain the optimal solution of NUM problem in multi-path situation, and also prove the effectiveness of our utility function. Then design wCompound on the basis of these conclusions.
- We implement wCompound algorithm in the Linux kernel, and carry out sufficient experiments on our testbed to verify its validity. The results prove that wCompound can always achieve fair bandwidth allocation on bottleneck links and high link utilization no matter competing with what kinds of background flows, which is better than some of the existing MPTCP algorithms.

The rest of the paper is organized as follows. Section II elaborates the related work in detail, including the basic principle of CTCP, and some existing and recently proposed research of multipath congestion control. Section III analyzes the model of wCompound, carries out the derivation on the weighted parameter and presents the wCompound algorithm. Section IV discusses the implementation of wCompound and evaluates its performance based on experimental results. Section V concludes the paper.

II. Related Work

A. Basic Idea and Relative Studies of Compound TCP

Compound TCP [2] is a loss-and-delay-based TCP variant proposed for HSLD environment. CTCP inherits from TCP Vegas [7], and combines two modes of High-Speed TCP (HSTCP) [8] and NewReno [9] to meet the bandwidth utilization over high-BDP networks. CTCP compares α to the estimated Δ to determine the increase or decrease of congestion window every round, where α is a predefined constant and Δ is the backlog which should be calculated every round. CTCP tracks the minimal Round Trip Time (RTT) observed so far, so called RTT_{min}, and updates the variable Δ during the congestion avoidance phase as Eq. (1):

$$\Delta = (W/RTT_{min} - W/RTT) \cdot RTT_{min}, \tag{1}$$

When Δ exceeds α, CTCP will gracefully reduce W_{fast} based on predefined ζ, and add W_{fast} to the final congestion window W, as shown in Eq. (2) and Eq. (4). Where W_{fast} is a smooth movement from HSTCP fastmode to NewReno slowmode. When Δ is less than α, which means there is neither congestion nor queue increase or packet loss occurs, CTCP will increase W as shown in Eq. (3) and Eq. (4). After the occurrence of packet loss, W is multiplicatively decreased as Eq. (5):

$$W_{fast}(n+1) = W_{fast}(n) - \zeta \cdot \Delta, \tag{2}$$

$$W_{fast}(n+1) = W_{fast}(n) + (\gamma \cdot W(n)^k - 1), \tag{3}$$

$$W(n+1) = W_{reno}(n+1) + W_{fast}(n+1). \tag{4}$$

$$W(n+1) = W_{reno}(n+1) \cdot (1-\beta). \tag{5}$$

Tunable parameters k, γ and β can be adjusted according to practical situation to provide desired scalability, smoothness and responsiveness. The suggested values given by [2] are 0.75, 0.125 and 0.5. Parameter ζ only affects the decreasing degree of window when early congestion occurs, $\zeta > 0$.

An important change to CTCP algorithm is that it combines both loss and delay based congestion avoidance approaches. The delay-based approach helps CTCP make use of the free bandwidth more efficiently and proactively slacken data rate when a link queue is sensed, and the loss-based approach ensures the lower bound of CTCP's throughput no less than TCP Reno. These two approaches are complementary.

Although CTCP has been proposed many years ago, it still plays an essential role in understanding and improving the network performance. Literatures [10] and [11] discuss the application of CTCP in WiFi, and mainly pay attention to improving its performance and guaranteeing throughput fairness. Literature [12] points out that in the current wireless local area network (WLAN) environment, CUBIC-TCP used in Linux operation system (OS) and CTCP used in Microsoft Windows OS are selected by most terminals as their TCP protocol, and introduces a throughput control method to guarantee that the terminals using CUBIC-TCP and the terminals using CTCP attain the same total throughput.

B. Existing Approaches and Their Limitations

Improving network transmission has been a long-term research interest all the time, and congestion control (CC) is one of its important research issues. In this paper, we focuse on CC. And we first introduce some classic and latest multipath congestion control schemes in this section to lay the groundwork for our algorithm.

We first introduce MPTCP. MPTCP is a fully backward-compatible extension of TCP. There are three constraints must be satisfied when designing MPTCP [13], [14]:

(i) Performance enhancement. MPTCP should at least perform as good as a single-path TCP running on the best available path.

(ii) Load balancing. The load of a MPTCP connection should be able to be shifted reasonably among all subflows. Namely the traffic on congested paths should be moved to non-congested paths as much as possible, but on the premise of meeting goal (i) and goal (iii).

(iii) Do no harm. MPTCP flow should ensure the fairness of bandwidth usage and be friendly to single-path flow. When several MPTCP subflows and single-path flows sharing the same bottleneck, on this bottleneck, the total bandwidth obtained by MPTCP subflows should be very similar to that of a single-path flow.

Congestion control is a key research content of MPTCP, and many congestion control algorithms (CCAs) have been proposed so far. Similar to TCP, MPTCP CCAs can also be divided into two main types:

Loss-based congestion control algorithms: for example, Linked increase algorithm (LIA) [14], OLIA [15], BALIA [16], D-LIA [17] and Couple+ [18]–[20], where packet loss plays a directive role in their load balancing and congestion control. However, in the phase of congestion avoidance, these algorithms adopt a very conservative approach to update their window. Such as LIA, which is the currently adopted MPTCP CCA, limits the increment of the total window of all subflows to no more than one packet in every round, which is very inefficient in HSLD network. It will require an unreasonable time to expand its window to a proper value [2].

Delay-based congestion control algorithms: such as weighted Vegas (wVegas) [21], mVegas [22] and mVeno [23], which adjust the transmission rates according to the variation of packet queuing delay. wVegas has been proven to behave less efficiently on paths with high BDP. However, these algorithms encounter problems when competing with aggressive flows, especially when the majority flows are loss-based, the throughput of delay-based algorithms tends to be severely compressed. This is because that delay-based algorithms are very sensitive to changes in network conditions, and once bottleneck queue is built, they will reduce their sending rate. Although this behavior can avoid self-induced packet losses, it also provides an incentive for loss-based flows to increase their rate. And the more loss-based flows increase their rate, the more delay-based flows will give in. Consequently loss-based flows will continue taking up bandwidth until it severely exceeds the fair share, while the rate of delay-based flows will keep getting lower and even be exhausted.

There are also schemes based on non-traditional methods. Inspired by BBR, several multipath BBR implementations have been proposed recently. In literature [24], Nguyen *et al.* extend BBR to MPTCP, in order to adapt to the situation of more commonly packet loss in wireless networks. But in their work, the single-path BBR is just simply applied to multipath scenarios without any changes. Zhang *et al.* [25] introduce another multipath BBR scheme named Delay-BBR, which sets a certain threshold for the round trip delay signal to avoid the bottleneck buffer being fully occupied. However, these two approaches, i.e., [24] and [25], are both uncoupled and might cause serious fairness issues for single-path flows. Han *et al.* [26] emphasize the improvement on fairness, and introduce a multiplication factor related to the bandwidth of each subflow. Zhu *et al.* [27] design weighted BBR (wBBR) based on the Congestion Equality Principle [21]. The algorithms introduced in [26], [27] are coupled, but attach too much importance to fairness. As a result, they can not satisfy constraint (i) and may even achieve lower throughput than single-path flows. Wei *et al.* [28] proposed BBR-based congestion control and data scheduling under the consideration of bottleneck fairness. Currently the vast majority of BBR-based multipath implementations are not widely used, and also do not solve the built-in problem of BBR mechanism (see Section I).

At present, there are still large vacancies in the research of using MPTCP to improve the performance of HSLD networks. Most of the MPTCP CCAs described previously mainly emphasize the goal of fairness, but incapable of meeting the goal of improving performance in most scenarios [29]. And such feature does not fit the needs of HSLD networks.

III. The Proposed wCompound Algorithm

We provide a loss-and-delay-based congestion control algorithm for MPTCP that is adapted to HSLD network, named wCompound. As the name indicates, wCompound is originated from CTCP. It controls the sending rate of each subflow by assigning various weights to different subflows and affecting their thresholds. With reasonably designing the weight allocation algorithm, all the subflows can be effectively coupled and thus improve the overall performance of MPTCP in HSLD network. In this section, we first introduce the utility function of wCompound, then the derivation of the weighted parameter of each subflow, and finally we design the wCompound algorithm.

Table I summarizes the mathematical notation used in our model and derivation.

TABLE I
Mathematical notation used in model and derivation

	Description
S	A set of all flows in a network
s	A flow in set S, $s \in S$
R_s	A set of all subflows of flow s
r	A subflow in set R_s, $r \in R_s$
L	A set of all available links in a network
l	A link in set L, $l \in L$
c	A set of the capacity of link l, for $l \in L$
A	A routing matrix describing the relationship between L and R
B	A matrix describing the relationship between s and R_s
λ	Lagrange multiplier associated with link l, for $l \in L$ ($\lambda_l \geq 0$)
ε	Search step size ($\varepsilon > 0$)
t	Iteration index
$win_{s,r}$	Sending window of flow s on subflow r
win_{max}	The maximum sending window during the transmission (for single-path situation)
win_{max_r}	The maximum sending window of flow s on subflow r during the transmission (for multipath situation)
$x_{s,r}$	Transmission rate of flow s on subflow r
x	Represents the vector $(x_{s,r}, s \in S, r \in R_s)$
y_s	Total rate of flow s (calculated as $y_s = \sum_{r\in R_s} x_{s,r}$)
y	Represents the vector $(y_s, s \in S)$
$diff$	Number of backlogged packets in the queue (for single-path situation)
$diff_{s,r}$	Number of backlogged packets in the queue of flow s on subflow r (for multipath situation)
$\alpha_{s,r}$	Expected backlogged packets in link queues of flow s on subflow r
α_s	Total backlogged packets in link queues of flow s
γ, β, k, ξ	Tunable parameters which affect the scalability, smoothness and responsiveness of CTCP and wCompound
$\dot{x_s}$	Average reduction factor (needs to be discussed in different cases)
$E\left[\eta_i\right]$	Average decrease factor when no packet loss occurs (for single-path situation)
$E_s\left[\eta_i\right]$	Average decrease factor for flow s when no packet loss occurs (for multipath situation)
$P_{diff<m}$	Probability that $diff$ is less than m (where m is the threshold selected by a certain subflow)
q_r	Path price of subflow r
q_s	The same price owned by all working subflows of flow s at the equilibrium point
$k_{s,r}$	Weight of flow s on subflow r
C, C_s, $C_s{}'$	Constants generated in the derivation process, and have no impact on the final results
$(*)^+$, $[*]^+$	Projection onto the nonnegative orthant, synonymous with $max(*, 0)$
$U(x)$	Utility function of a flow when its data rate is x (it is built for single-path CTCP in this paper)
$U_s(y_s)$	Utility function of flow s when its total data rate is y_s (it is built for multipath wCompound in this paper)
$U'{}_s(y_s)$	First derivative of $U_s(y_s)$

A. *Extending CTCP from Single-path to Multi-path*

In multipath transmission, we model the network is shared by a set $S = \{1, \cdots, s\}$ of flows, and each flow $s \in S$ consists of a set of subflows R_s. Every subflow $r \in R_s$ may take a different route, and maintains its own sending window $win_{s,r}$. The transmission rate of flow s on subflow r is set as $x_{s,r}$, so the total rate y_s of flow s is the sum of the rates of all subflows, we have $y_s = \sum_{r\in R_s} x_{s,r}$.

Since wCompound is a multipath transmission scheme derived from CTCP, it also adopts CTCP's congestion control method. The sending Window of CTCP is controlled by two parts: loss-based $cwnd$ and delay-based $dwnd$. The sending window (hereafter called win) is calculated as Eq. (6):

$$win = min(awnd, cwnd + dwnd), \tag{6}$$

where $awnd$ is the advertised window from the receiver.

From Eq. (6) we know that CTCP can send $cwnd + dwnd$ packets every round, hence the increment of $cwnd$ on receiving each ACK should also be changed:

$$cwnd = cwnd + \frac{1}{cwnd + dwnd}. \tag{7}$$

CTCP maintains a state variable named $baseRTT$, which is used to record the minimum RTT observed so far. So in the last round, the number of backlogged packets $diff$ in queue is calculated as Eq. (8):

$$diff = (\frac{win}{baseRTT} - \frac{win}{RTT}) \cdot baseRTT, \tag{8}$$

where $win/baseRTT$ expresses the estimation of throughput in the case of not overrunning the network capacity, i.e. the expected throughput. win/RTT expresses the throughput actually obtained. Subtract the actual throughput from the expected throughput and multiply by $baseRTT$, we can have the number of backlogged packets in bottleneck queue.

Then, compare $diff$ with the threshold value α. If $diff \geq \alpha$, the path is considered to be busy, which is a cue that network congestion has occurred so far, so the delay-based component will reduce its window appropriately to alleviate the network congestion. Otherwise, the path is considered to be under-utilized.

On each subflow, wCompound works in the same way as CTCP. wCompound similarly calculates $diff$ according to Eq. (8) and compared it with the corresponding threshold. However, the threshold mentioned here is different from the fixed empirical value set in CTCP, but a dynamic threshold which can be adaptively adjusted according to current network state. Here we set the expected number of backlogged packets on subflow r as $\alpha_{s,r}$, and wCompound's behaviour on each subflow is as follows: If $diff < \alpha_{s,r}$ and no packet loss occurs, subflow r increases its sending window $win_{s,r}$ according to:

$$win_{s,r}(t+1) = win_{s,r}(t) + \gamma \cdot win_{s,r}(t)^k. \tag{9}$$

Otherwise, if $diff > \alpha_{s,r}$, $win_{s,r}$ is reduced by:

$$win_{s,r}(t+1) = win_{s,r}(t) - \xi \cdot diff{+}1. \tag{10}$$

When there is a loss due to congestion, $win_{s,r}$ is multiplicatively decreased as:

$$win_{s,r}(t+1) = (1-\beta) \cdot win_{s,r}(t). \tag{11}$$

Accordingly, the delay-based component $dwnd$ needs to be complementary to $cwnd$, so $dwnd$ is adjusted according to Eq. (12).

$$dwnd(t{+}1) = \begin{cases} dwnd(t){+}(\gamma \cdot win(t)^k{-}1)^+, & diff{<}\alpha_{s,r} \\ (dwnd(t){-}\xi \cdot diff)^+, & diff{>}\alpha_{s,r} \\ ((1{-}\beta) \cdot win(t){-}cwnd/2)^+, & \text{packet\_loss} \end{cases} \tag{12}$$

where $(*)^+$ is defined as the projection of $*$ onto the nonnegative orthant. Parameters β, γ and k are adjustable to provide ideal scalability, smoothness and responsiveness. And parameter ξ defines the decreasing degree of the delay-based component when early congestion is detected, here we need to make sure that ξ is positive.

For the congestion avoidance phase, assuming no packet loss occurs, the average reduction factor is: $\dot{x_s} = E[\eta_i]$.

The average reduction factor in the event of congestion loss is: $\dot{x_s} = -\beta \cdot x_s$.

$P_{diff<m}$ represents the probability that $diff$ is less than m. According to literature [23], we define it as:

$$P_{diff<m} = min\left(1, \frac{m \cdot RTT}{(RTT - baseRTT) \cdot win_{max}}\right), \tag{13}$$

where win_{max} stands for the maximum window size during the transmission.

Therefore, the source will receives $win \cdot P_{diff<m}$ positive acknowledgments or $win{\cdot}P_{diff>m}$ negative acknowledgments per round. According to Eq. (9) and Eq. (10), positive acknowledgment and negative acknowledgment correspond to different multiplicative coefficients respectively. So put the expected increment and decrement together, the average decrease factor $E[\eta_i]$ is calculated as:

$$\begin{aligned} E[\eta_i] &= \gamma \cdot win^k \cdot P_{diff<m} + (-\xi \cdot diff) \cdot P_{diff>m} \\ &= \frac{\left(\gamma \cdot win^k {+} \xi \cdot diff\right) \cdot m \cdot RTT}{(RTT - baseRTT) \cdot win_{max}} - \xi \cdot diff. \end{aligned} \tag{14}$$

B. Network Utility Maximization Model

As proved by Kunniyur *et al.* [30], TCP's behavior could be modeled by a framework based on the utility maximization approach when taking no account of slow start phase. In the modeling of this paper, we also do not consider slow start and the impact of timeout, and design the utility function according to the framework and matching method proposed in literature [31].

Hurley *et al.* [32] propose an utility function applicable to TCP-type congestion avoidance. They start their analysis with a stochastic model and an ordinary differential equation can be derived from this model. Literature [32] also uses a discrete-time model since the events in TCP usually occur at packet level, it proves that the utility function applicable to TCP-type congestion avoidance is in the form of $log\frac{x}{1+x}$. When x is large:

$$log\frac{x}{1+x} = log\frac{1}{1+\frac{1}{x}} \approx log\left(1 - \frac{1}{x}\right) \approx -\frac{1}{x}. \tag{15}$$

This approach does not model TCP precisely, since it ignores partial TCP phases such as slow-start and timeout. But $\frac{1}{x}$ in Eq. (15) still reveals the negative correlation between throughput and the product of RTT and loss probability.

Reference to the existing research [33], we can construct the utility function of CTCP:

$$\begin{aligned} U(x) = &\left(\frac{E[\eta_i]}{\beta} + x\right)[log\,(E[\eta_i] {+} \beta x) - 1] \\ &- x\,[log\,(\beta x) - 1] + C, \end{aligned} \tag{16}$$

where C is a constant, and has no effect on the subsequent derivation.

According to Eq. (16) for single-path situation, replace rate x with rate y_s, similarly, the utility function of wCompound can be constructed as:

$$\begin{aligned} U_s(y_s) = &\left(\frac{E_s[\eta_i]}{\beta} + y_s\right)[log\,(E_s[\eta_i] {+} \beta y_s) - 1] \\ &- y_s\,[log\,(\beta y_s) - 1] + C_s, \end{aligned} \tag{17}$$

where C_s is a constant.

When modeling a network, suppose it contains finite links accessed by finite users. The link set is described as L, the user set is described as S, and all links have finite capacities $c = (c_l, l \in L)$. Matrix A describes the relationship between link l and path r. Specifically, $A = (a_{l,r}, l \in L, r \in R)$, and $a_{l,r} = 1$ if $l \in L_r$, or else $a_{l,r} = 0$, where a path $r \in R$ is defined as the subset $L_r \in L$. Each flow $s \in S$ is associated with a subset $R_s \in R$, matrix B describes the relationship between flow s and R_s. Specifically, $B = (b_{s,r}, s \in S, r \in R)$, and $b_{s,r} = 1$ if $r \in R_s$, or else $b_{s,r} = 0$. Represent the vector $(x_{s,r}, s \in S, r \in R_s)$ by x, and the vector $(y_s, s \in S)$ by y. As described in [34] by Kelly, the network bandwidth allocation problem can be formulated as a NUM problem. The main purpose of NUM is to maximize the aggregate utility of users receiving bandwidth on the premise of limited link capacity, which can be formulated as:

$$\begin{aligned} &maximize \sum\nolimits_{s \in S} U_s(y_s) \\ &s.t. \quad y = Bx \\ &\qquad\quad Ax \le c. \end{aligned} \tag{18}$$

In order to solve the NUM problem effectively, the idea of convex optimization is used to obtain approximate solutions.

A NUM problem is considered to be solvable if it can be classified as a convex optimization problem. Moreover, the local optimal solution of the convex optimization problem is also its global optimal solution, and this excellent characteristic makes it easier for us to solve the NUM problem. There are some necessary conditions for convex optimization [35]: The constructed utility function $U(x)$ must be non-decreasing smooth function, strictly concave, and differentiable in the range of $x > 0$.

For the objective function is strictly concave for y and the feasible region is compact [21], the existence and uniqueness of the optimal solution for y can be easily proved. Nevertheless, the optimal solution for x is not necessarily unique. Consider the Lagrangian function [21]:

$$\begin{aligned} L(x,\lambda) :&= \sum_{s\in S} U_s(y_s) + \sum_{l\in L}\lambda_l\left(c_l - \sum_{r\in R} a_{l,r}x_r\right) \\ &= \sum_{s\in S} U_s(y_s) - \sum_{l\in L}\sum_{r\in R}\lambda_l a_{l,r}x_r + \sum_{l\in L}\lambda_l c_l \\ &= \sum_{s\in S} U_s(y_s) - \sum_{r\in R} q_r x_r + \lambda \mathrm{c}^T \\ &= \sum_{s\in S} U_s(y_s) - \sum_{s\in S}\sum_{r\in R} b_{s,r} q_r x_r + \lambda \mathrm{c}^T \\ &= \sum_{s\in S} U_s\left(\sum_{r\in R_s} x_{s,r}\right) - \sum_{r\in R_s} q_r x_{s,r} + \lambda \mathrm{c}^T, \end{aligned} \tag{19}$$

where

$$q_r = \sum_{l\in L}\lambda_l \alpha_{l,r}, \tag{20}$$

and parameter $\lambda = (\lambda_l, l \in L)$ is a introduced Lagrange multiplier used for combining the constraint conditions with the original objective function. We can regard λ_l as the congestion mark or price of link l. Suppose subflow r is constituted by several links, which is represented by matrix A. Thus q_r is the path price of subflow r.

Define:

$$L_s(\lambda) := \max_{x_{s,r}\geq 0} U_s\left(\sum_{r\in R_s} x_{s,r}\right) - \sum_{r\in R} q_r x_{s,r}, \tag{21}$$

$$D(\lambda) := \sum_{s\in S} L_s(\lambda) + \lambda \mathrm{c}^T. \tag{22}$$

According to the Lagrangian dual method, the original problem (18) can be solved by transforming it into its dual problem, which can be expressed as:

$$\underset{\lambda\geq 0}{min}\, D(\lambda). \tag{23}$$

So far we decompose the original objective function (18) into a main problem (23) and a set of subproblems (21). Every subproblem has an local optimal solution which is only related to the local knowledge q_r of a path ($r \in R$).

On the source side, given the premise of λ, flow s obtains the optimal solution $x^*_{s,r}(\lambda)$ by solving (21). Then $x^*_{s,r}(\lambda)$ is broadcasted to other links, and the price λ_l of link l is adjusted in the opposite direction to the gradient of Eq. (22) as follow, and then announced to flows:

$$\lambda_l(t+1) = \left[\lambda_l(t) - \varepsilon\left(c_l - \sum_{r:l\in L_r} x^*_{s,r}(\lambda)\right)\right]^+, \tag{24}$$

where ε represents the search step size, t represents the iteration index. These steps are executed cyclically, until they eventually converge to the dual optimal solution λ^*, thus the corresponding optimal solutions of the original problem is $x^*(\lambda^*)$.

Next, we need to optimize the foregoing iterative process with the consideration of the particularity of multipath congestion control. There are some problems deserve consideration: First, at each step of the iterative process, every flow turns off all its subflows except for the least congested one, which is not applicable to real network. Second, when a flow receives the same price from multiple subflows, it cannot determine a unique solution of Eq. (21).

To solve the above problems, we choose an iteration method for calculating approximate value of the original problem. It is of more feasibility to calculate an approximate solution $x_{s,r}(\lambda)$ than calculating the optimal solution $x^*_{s,r}(\lambda)$. Each iteration step starts from current rates and advance a distance in the direction of the gradient of

$$G_s(x) := U_s\left(\sum_{r\in R_s} x_{s,r}\right) - \sum_{r\in R_s} q_r x_{s,r} \tag{25}$$

to get the new rates, and then broadcast them to links.

Take the derivative of Eq. (25):

$$\frac{\partial G_s(x)}{\partial x_{s,r}} = U'_s(y_s) - q_r. \tag{26}$$

Karush-Kuhn-Tucker (KKT) conditions indicate the essential conditions which should be simultaneously satisfied for the optimal solution of (21):

$$\frac{\partial G_s(x)}{\partial x_{s,i}} = U'_s\left(\sum_{r=1}^{n} x_{s,r}\right) - q_i \leq 0, \tag{27}$$

$$\frac{\partial G_s(x)}{\partial x_{s,i}} x_{s,i} = 0, \tag{28}$$

$$x_{s,i} \geq 0, \tag{29}$$

where $i = 1, 2, \cdots, n$. First, suppose $q_1 = q_2 = \cdots = q_n$. For there is at least one subflow with a positive rate, denoted as j, from (28) we have $\partial G_s(x)/\partial x_{s,j} = 0$, thus $U'_s\left(\sum_{r=1}^{n} x_{s,r}\right) - q_j = 0$. Second, suppose $q_1 = \cdots = q_a < q_{a+1} \leq \cdots \leq q_n$. If we have subflow j ($j > a$) with $x_{s,j} \neq 0$, then we have $\partial G_s(x)/\partial x_{s,j} = 0$, and from (27) we have $U'_s\left(\sum_{r=1}^{n} x_{s,r}\right) - q_j = 0$. So there is $U'_s\left(\sum_{r=1}^{n} x_{s,r}\right) = q_j > q_1 \geq U'_s\left(\sum_{r=1}^{n} x_{s,r}\right)$, which leads to a contradiction, thus this assumption is invalid. All in all, the optimal solution $x^*_{s,r}(\lambda)$ of (21) is proved to satisfy:

$$U'_s\left(\sum_{r=1}^{a} x^*_{s,r}(\lambda)\right) - q_1 = 0, \tag{30}$$

$$x^*_{s,r}(\lambda) = 0, r = a+1, \cdots, n. \tag{31}$$

This result indicates that a flow strives to maximize its utility by giving up expensive paths and only choose the cheapest one. And as more traffic is injected into a path, its price will increase accordingly, and vice versa. When every path shares the same price, which means the congested degree perceived on all available paths is equal, the requirement of Eq. (18) is fulfilled perfectly. And we can achieve this congestion equality by traffic shifting.

As to the gradient direction mentioned in Eq. (25), since the rate tends to increase on less congested paths but decrease on more congested paths, the convergence of the iterative process is still guaranteed while obtaining the approximate solution. As t approaches infinity, we will have the dual optimal solution λ^* and the primal optimal solution $x^*(\lambda^*)$.

According to Eq. (26), flow s updates its rates by

$$x_{s,r}(t+1) = \left[x_{s,r}(t) + \theta(U'_s(y_s) - q_r)\right]^+, r \in R_s \tag{32}$$

at each iteration step, where θ is a positive step size. And the physical significance of $U'_s(y_s)$ is the expected path price of flow s, where all its subflows are merged into a virtual single path with the rate of y_s.

According to Eq. (32), if $U'_s(y_s)$ is lower than the current path price q_r, the sending rate will decrease, or else increase. Consequently, traffic on the congested paths is always shifted to non-congested paths, so each path tends to be equally congested. Therefore, the Congestion Equality Principle [21] is satisfied, which means the congestion degree of each subflow should be as consistent as possible, thus to achieve effective load balancing and fair sharing of network resources between subflows. Accordingly, replace the optimal solution $x^*_{s,r}(\lambda)$ in Eq. (24) with the approximate solution $x_{s,r}(\lambda)$:

$$\lambda_l(t+1) = \left[\lambda_l(t) - \varepsilon\left(c_l - \sum_{r:l\in L_r} x_{s,r}(\lambda)\right)\right]^+. \tag{33}$$

So far, Eq. (32) and Eq. (33) together constitute the approximate iterative algorithm for solving (23).

Next we want to verify whether the utility function constructed earlier satisfies these necessary conditions for convex optimization. Take the derivative of Eq. (17):

$$\begin{aligned} U'_s(y_s) &= log\left(E_s\left[\eta_i\right] + \beta y_s\right) - log\left(\beta y_s\right) \\ &= log\left(\frac{E_s\left[\eta_i\right]}{\beta y_s} + 1\right). \end{aligned} \tag{34}$$

According to Eq. (34), obviously, the constructed utility function $U_s(y_s)$ is increasing, twice continuously differentiable and strictly concave in the nonnegative domain, which satisfies all the necessary conditions for convex optimization. As $E_s\left[\eta_i\right] \geq 0$, we have $U'_s(y_s) \geq 0$.

C. Design of wCompound Algorithm

Next, it is known that $diff$ will converge to α at the equilibrium point, we set $q_r = RTT - baseRTT$, $x_{s,r} = win/RTT$, and replace $diff$ with $\alpha_{s,r}$, so there is:

$$x_{s,r} = \frac{\alpha_{s,r}}{q_r}, r \in R_s. \tag{35}$$

At this point, the prices of all working subflows of flow s are the same, represented as q_s, and there is:

$$y_s = \sum_{r\in R_s} x_{s,r} = \sum_{r\in R_s} \frac{\alpha_{s,r}}{q_r} = \frac{1}{q_s}\sum_{r\in R_s} \alpha_{s,r} = \frac{\alpha_s}{q_s}. \tag{36}$$

Based on the approximate iterative algorithm and the necessary conditions satisfied by the optimal solution, we can construct the utility function of wCompound in another form. From Eq. (30), Eq. (31) and Eq. (36), we have:

$$U'_s(y_s) = \sum_{r\in R_s} q_r = q_s = \frac{\alpha_s}{y_s}. \tag{37}$$

Perform an inversion operation via Eq. (37), and another expression of $U_s(y_s)$ can be obtained as:

$$U_s(y_s) = \alpha_s log y_s + C_s{}', \tag{38}$$

where $C_s{}'$ is a constant. It can be observed that Eq. (38) has the same characteristics as Eq. (17), they are both increasing, strictly concave and twice continuously differentiable. So take the derivative of the right half of both Eq. (34) and Eq. (37):

$$\begin{gathered} \left[\frac{\alpha_s}{y_s}\right]' = \left[log\left(\frac{E_s\left[\eta_i\right]}{\beta y_s} + 1\right)\right]', \\ \alpha_s = \frac{E_s\left[\eta_i\right]}{\left(\frac{E_s\left[\eta_i\right]}{\beta y_s} + 1\right)\cdot\beta} = \frac{E_s\left[\eta_i\right]\cdot y_s}{E_s\left[\eta_i\right] + \beta y_s}, \end{gathered} \tag{39}$$

where $E_s\left[\eta_i\right]$ represents the average decrease factor of flow s. Let $diff_{s,i}$ denote the $diff$ of subflow i, so $E_s\left[\eta_i\right]$ can be calculated as:

$$\begin{aligned} E_s\left[\eta_i\right] &= \sum_{i\in R_s} E_{s,i}\left[\eta_i\right] \\ &= \sum_{i\in R_s}\left\{\frac{\left(\gamma\cdot win_{s,i}{}^k + \xi\cdot diff_{s,i}\right)\cdot\alpha_{s,i}\cdot RTT_{s,i}}{\left(RTT_{s,i} - baseRTT_{s,i}\right)\cdot win_{max\_i}}\right. \\ &\quad \left. -\xi\cdot diff_{s,i}\right\}. \end{aligned} \tag{40}$$

Set $\theta = x_{s,r}(t)/q_r$ and substitute it into Eq. (32), so flow s updates its rate according to

$$x_{s,r}\left(t+1\right) = \frac{U'_s(y_s)}{q_r} x_{s,r}\left(t\right), r \in R_s \tag{41}$$

every iteration step.

As explained earlier, $U'_s(y_s)$ is the expected path price, so Eq. (41) follows the Congestion Equality Principle. And we substitute Eq. (37) into Eq. (41) to get the formula used to update rate:

$$x_{s,r}\left(t+1\right) = \frac{\alpha_s}{q_r}\cdot\frac{x_{s,r}\left(t\right)}{y_s}, r \in R_s. \tag{42}$$

Set $k_{s,r}\left(t\right) = x_{s,r}\left(t\right)/y_s$. So the $\alpha_{s,r}$ assigned to subflow r can be solved as:

$$\alpha_{s,r} = k_{s,r}\cdot\alpha_s = \frac{x_{s,r}}{\sum\limits_{i\in R_s} x_{s,i}}\cdot\frac{E_s\left[\eta_i\right]\cdot y_s}{E_s\left[\eta_i\right] + \beta y_s}, \tag{43}$$

Specifically, $k_{s,r}$ determines the share of weight on a certain subflow, while α_s determines the weight that the whole

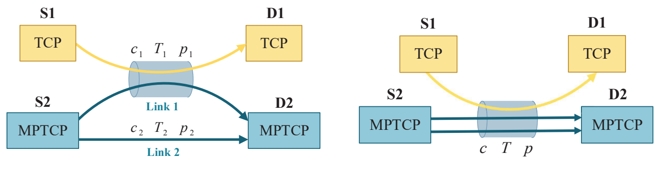
Algorithm 1: Congestion Avoidance Phase of wCompound

```
1  Initialization :
2  begin
3      for $r \in R_s$ do
4          Initialize $\alpha_{s,r}$, $x_{s,r}$;
5  Weight Adjustment( ) :
6  begin
7      $k_{s,r} \leftarrow x_{s,r}/y_s$;
8      Calculate $E_s[\eta_i]$ and $E_{s,i}[\eta_i]$ according to Eq.(40);
9      $\alpha_s \leftarrow (E_s[\eta_i] \times y_s) / (E_s[\eta_i] + \beta \times y_s)$;
10     return $k_{s,r} \times \alpha_s$;
11 Upon receiving each ACK :
12 begin
13     $rtt_{s,r} \leftarrow \sum_r sampled\_rtt_{s,r}/sampled\_num_{s,r}$;
14     $baseRTT_{s,r} \leftarrow min\{sampled\_rtt_{s,r}\}$;
15     $baseRTT \leftarrow min\{baseRTT_{s,r}\}$;
16     $win_{max\_r} \leftarrow max\{win_{s,r}\}$;
17     $diff_{s,r} \leftarrow win_{s,r} \times (rtt_{s,r} - baseRTT_{s,r}) / rtt_{s,r}$;
18     // calculate weights and alphas
19     $x_{s,r} \leftarrow win_{s,r}/rtt_{s,r}$;
20     $y_s \leftarrow \sum_{r \in R_s} x_{s,r}$;
21     $\alpha_{s,r} \leftarrow WeightAdjustment()$;
22     // window adjustment
23     $cwnd_{s,r}++$;
24     if $diff < \alpha_{s,r}$ then
25         $dwnd_{s,r} += (\gamma \cdot win_{s,r}{}^k - 1)^+$;
26         $win_{s,r} = cwnd_{s,r} + dwnd_{s,r}$;
27     else
28         if $dwnd_{s,r} > \xi \cdot diff$ then
29             $dwnd_{s,r} -= \xi \cdot diff$;
30         else
31             $dwnd_{s,r} = 0$
32         $win_{s,r} = cwnd_{s,r} + dwnd_{s,r}$;
```

MPTCP connection should be assigned under current network state.

It should be noted that $diff_{s,r}$, $E_s[\eta_i]$ and $\alpha_{s,r}$ need to be calculated in each round. Here, the weight $k_{s,r}$ quantifies the level of bandwidth competitive intensity. The less congested the subflow is, the greater the weight is allocated to this subflow, which will also lead to the increase of competitive intensity, thus affect the following weight allocation, and vice versa. Eventually, an equilibrium point will be reached where each flow is equally congested.

This subsection introduces the methods for calculating the weighted parameter $\alpha_{s,r}$ for each subflow r, and the steps outlined in this subsection form the algorithm of wCompound. The pseudo-code is given by Algorithm 1. The initial value of $\alpha_{s,r}$ is set to 6, but has little effect on the subsequent operation.



(a) Topology 1 (b) Topology 2

Fig. 1. The network topologies

IV. Performance Evaluation

We validate the proposed wCompound algorithm experimentally in our testbed. We implement wCompound as Linux model, and this scheme only need to be executed on the server. Our deployed testbed consists of two file servers, two routers and two clients, both the servers and the clients are running on Linux ubuntu 16.04 OS with MPTCP kernel version 4.19.142 [36]. We use these components to build two classic network topologies as shown in Fig. 1(a) and Fig. 1(b), where every MPTCP connection consists of two subflows, and c represents the bandwidth of link, T represents the propagation delay, p represents the random packet loss rate. Then compare the performance of wCompound in these scenarios with four existing algorithms: LIA [14], OLIA [15], Balia [16] and wVegas [21], which have already been implemented in the kernel.

A. Network Utilization

We adopt the topology of Fig. 1(a) to evaluate how effectively can wCompound utilize the available bandwidth in HSLD scenario. Specifically, we take no account of the impact of random packet loss, and the parameters are set as: $c_1 = c_2 = 150Mbps$, $T_1 = T_2 = 80ms$, $p_1 = p_2 = 0$.

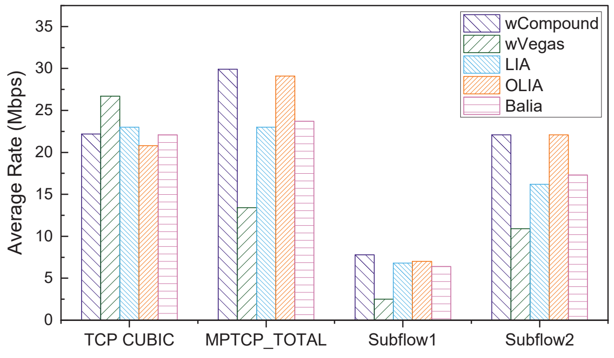


Fig. 2. The average rates of CUBIC and two subflows

In the first set of experiments, we run 5 TCP CUBIC flows and 5 MPTCP flows simultaneously. In isolation TCP flow and subflow1 run concurrently on Link 1, resulting in a higher congestion level of Link 1 than Link 2. The aggregated data is shown in Fig. 2, where $MPTCP\_TOTAL$ stands for the sum of the rates of subflow1 and subflow2. For wVegas, as shown in the figure, when TCP flows adopt loss-based congestion control method, it will preempt almost all the resources of Link 1, and subsequently affect subflow2 on the less congested

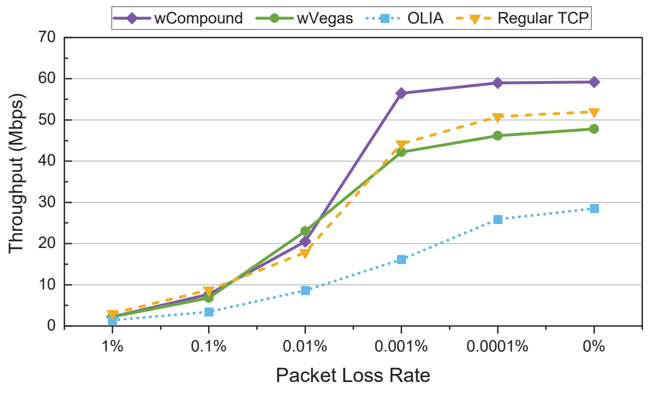


Fig. 3. Throughputs under different packet loss rates

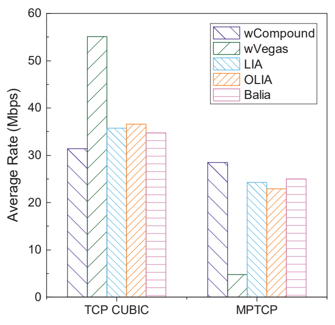


Fig. 4. Sharing bottleneck link with loss-based TCP flows

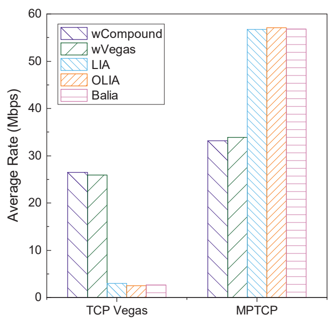


Fig. 5. Sharing bottleneck link with delay-based TCP flows

link (Link 2), thus resulting in severe compression of the total rate of wVegas. This is because the throughput maximization is possible to the original MPTCP only when all its available paths share a common bottleneck [37]. Among all the tested MPTCP algorithms, wCompound achieves the highest total rate, and shifts most of its traffic to the less congested Link 2. Meanwhile, the average rate of TCP with wCompound basically reaches the same level of LIA and Balia, which means wCompound achieves the throughput enhancement by utilizing the free available bandwidth more sufficiently rather than occupying the bandwidth of regular TCP. By contrast, although OLIA achieves the same effect as wCompound in load balancing, it does not utilize the bandwidth which is not efficiently used by TCP on Link 1 as fully as wCompound does.

We then conduct another set of experiments to further verify the ability of wCompound to utilize bandwidth when there are random packet losses in the network. We add random packet losses to the network and do not consider background flows, then compare wCompound with wVegas and OLIA. We vary the loss rate p from 10^{-2} to 10^{-6} and run wCompound, wVegas, OLIA and regular TCP flows respectively. The aggregated throughput is plotted in Fig. 3. As OLIA is a loss-based approach, its throughput is lower than the others when $p > 10^{-3}$, while the throughput of the other protocols is very close. However, as the packet loss rate decreasing from 10^{-3} to 0, the bandwidth utilization of wCompound is obviously higher than that of the other three protocols, and OLIA shows a marked insufficiency in network utilization.

B. Fairness to Different TCP Variants

To evaluate the fairness to different TCP variants, we adopt the topology shown in Fig. 1(b) that two wCompound subflows share the same bottleneck link with a regular TCP flow. Specifcally, we set $c = 300Mbps$, $T = 80ms$, $p = 0$. We run 5 TCP flows and 5 MPTCP flows simultaneously, and average the testing values as statistical sample.

In the third set of experiments, we divide the MPTCP algorithms involved into three groups: delay-based (wVegas), loss-based (LIA, OLIA, Balia) and wCompound. We test two scenarios, which take loss-based (CUBIC) flow and delay-based (Vegas) flow as the background flow respectively, so as to observe the performance of different MPTCP algorithms in competing with different types of TCP variants. The results are depicted in Fig. 4 and Fig. 5 respectively.

It can be seen from Fig. 4 that wVegas obtains much less throughput than CUBIC, this illustrates that MPTCP congestion control algorithms based on delay can hardly compete with loss-based TCP variants, and sadly it is difficult to remedy this defect by delay-based approaches themselves [2]. LIA, OLIA and Balia behave the same, they can maintain a relatively appropriate rate, but still far from catching up with CUBIC. When the background flows changes from CUBIC to Vegas, these two kinds of MPTCP approaches have exactly the opposite behavior to before, see Fig. 5. Delay-based MPTCP approaches show their fairness to delay-based TCP variants, while loss-based MPTCP flows show great aggressiveness and steal a lot of bandwidth from Vegas. However, compare to algorithms purely based on delay or loss, the proposed compound algorithm, namely wCompound, can always maintain a similar rate to regular TCP flows. wCompound provides better competitiveness with loss-based competing flows, and avoids the starvation caused by excessive concession. At the same time, under the mediation of delay-based component, wCompound does not behave aggressively and preempt additional bandwidth from others when coexisting with delay-based flows, which avoids the damage to the fairness of other flows. In a word, when sharing a common bottleneck link, wCompound is much fairer to different types of TCP variants than the existing MPTCP approaches.

V. Conclusions

In this paper, we presented a compound multipath CCA named wCompound to couple multiple subflows and help users transmit data more efficiently over HSLD networks, so as to provide a good QoS support. As the name indicates, wCompound is derived from CTCP. We aimed to effectively extend single-path CTCP to multipath to achieve coupled congestion control while satisfying the MPTCP constraints. The problem is analyzed based on the Network Utility Maximization model and solved by an approximate iterative approach, and we derived the weighted parameter and adaptive threshold of wCompound. As wCompound is an effective combination of loss-based method and delay-based method, it inherits the advantages from loss-based schemes and delay-based schemes to cope with multiple HSLD paths efficiently, and also ensure fair bandwidth allocation to different types of TCP variants at the shared bottleneck. We designed and implemented wCompound based on Linux kernel, the experimental results proved that wCompound outperforms the existing schemes and is a suitable transitional approach for HSLD networks.

ACKNOWLEDGMENT

This work is supported in part by the National Natural Science Foundation of China under Grants No. 61972371 and Youth Innovation Promotion Association of Chinese Academy of Sciences (CAS) under Grant No. Y202093.

REFERENCES

[1] M. A. Alrshah, M. A. Al-Maqri, and M. Othman, "Elastic-TCP: Flexible congestion control algorithm to adapt for high-BDP networks," *IEEE Systems Journal*, vol. 13, no. 2, pp. 1336–1346, 2019.

[2] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "A compound TCP approach for high-speed and long distance networks," in *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM)*, pp. 1–12, IEEE, 2006.

[3] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-based congestion control: Measuring bottleneck bandwidth and round-trip propagation time," *Queue*, vol. 14, no. 5, pp. 20–53, 2016.

[4] M. Hock, R. Bless, and M. Zitterbart, "Experimental evaluation of BBR congestion control," in *Proceedings of the 25th IEEE International Conference on Network Protocols (ICNP)*, pp. 1–10, 2017.

[5] 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.

[6] M. Mathis and J. Mahdavi, "Deprecating the TCP macroscopic model," *ACM SIGCOMM Computer Communication Review*, vol. 49, no. 5, pp. 63–68, 2019.

[7] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance," in *Proceedings of the 1994 ACM Conference on Communications Architectures, Protocols and Applications (SIGCOMM)*, pp. 24–35, ACM, 1994.

[8] S. Floyd, "HighSpeed TCP for large congestion windows," 2003. RFC 3649, IETF, Accessed on Apr., 2021.

[9] S. Floyd, T. Henderson, and A. Gurtov, "The NewReno modification to TCP's fast recovery algorithm," 2004. RFC 3782, IETF, Accessed on Apr., 2021.

[10] S. R. Pokhrel and S. Singh, "Compound TCP performance for industry 4.0 WiFi: A cognitive federated learning approach," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 3, pp. 2143–2151, 2021.

[11] S. R. Pokhrel and C. Williamson, "Modeling Compound TCP over WiFi for IoT," *IEEE/ACM Transactions on Networking*, vol. 26, no. 2, pp. 864–878, 2018.

[12] R. Tsurumi, M. Morita, H. Obata, C. Takano, and K. Ishida, "Throughput control method between different TCP variants based on SP-MAC over WLAN," in *Proceedings of the 2018 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)*, pp. 1–2, 2018.

[13] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath TCP," in *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation (NSDI)*, pp. 99–112, USENIX, 2011.

[14] C. Raiciu, M. Handley, and D. Wischik, "Coupled congestion control for multipath transport protocols," *IETF, RFC 6356*, October 2011.

[15] R. Khalili, N. Gast, M. Popovic, *et al.*, "Opportunistic linked-increases congestion control algorithm for MPTCP," *IEEE/ACM Transactions on Networking*, 2013.

[16] A. Walid, Q. Peng, J. Hwang, and S. Low, "Balanced linked adaptation congestion control algorithm for MPTCP," 2016. draft-walid-mptcp-congestion-control-04, IETF, Accessed on Apr., 2021.

[17] T. Lubna, I. Mahmud, and Y.-Z. Cho, "D-LIA: Dynamic congestion control algorithm for MPTCP," *ICT Express*, vol. 6, no. 4, pp. 263–268, 2020.

[18] K. Xue, J. Han, H. Zhang, K. Chen, and P. Hong, "Migrating unfairness among subflows in MPTCP with network coding for wired-wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 1, pp. 798–809, 2017.

[19] D. Ni, K. Xue, P. Hong, and S. Shen, "Fine-grained forward prediction based dynamic packet scheduling mechanism for multipath TCP in lossy networks," in *Proceedings of the 23rd International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–7, IEEE, 2014.

[20] D. Ni, K. Xue, P. Hong, H. Zhang, and H. Lu, "OCPS: Offset compensation based packet scheduling mechanism for multipath TCP," in *Proceedings of the 2015 IEEE International Conference on Communications (ICC)*, pp. 6187–6192, IEEE, 2015.

[21] Y. Cao, M. Xu, and X. Fu, "Delay-based congestion control for multipath TCP," in *Proceedings of the 20th IEEE International Conference on Network Protocols (ICNP)*, pp. 1–10, IEEE, 2012.

[22] P. L. Vo, T. A. Le, S. Lee, C. S. Hong, B. Kim, and H. Song, "Multi-path utility maximization and multi-path TCP design," *Journal of Parallel and Distributed Computing*, vol. 74, no. 1, pp. 1848–1857, 2014.

[23] P. Dong, J. Wang, J. Huang, and H. Wang, "Performance enhancement of multipath TCP for wireless communications with multiple radio interfaces," *IEEE Transactions on Communications*, vol. 64, no. 8, pp. 3456–3466, 2016.

[24] K. Nguyen, M. G. Kibria, K. Ishizu, F. Kojima, and H. Sekiya, "An evaluation of multipath TCP in lossy environment," in *Proceedings of the 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pp. 573–577, 2019.

[25] S. Zhang, W. Lei, W. Zhang, Y. Guan, and H. Li, "Congestion control and packet scheduling for multipath real time video streaming," *IEEE Access*, vol. 7, pp. 59758–59770, 2019.

[26] J. Han, K. Xue, Y. Xing, P. Hong, and D. S. Wei, "Measurement and redesign of BBR-based MPTCP," in *Proceedings of the 2019 ACM SIGCOMM Conference Posters and Demos*, pp. 75–77, 2019.

[27] T. Zhu, X. Qin, L. Chen, X. Chen, and G. Wei, "wBBR: A bottleneck estimation-based congestion control for multipath TCP," in *Proceedings of the 88th IEEE Vehicular Technology Conference (VTC-Fall)*, pp. 1–5, 2018.

[28] W. Wei, K. Xue, J. Han, Y. Xing, D. S. Wei, and P. Hong, "BBR-based congestion control and packet scheduling for bottleneck fairness considered multipath TCP in heterogeneous wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 1, pp. 914–927, 2021.

[29] R. Lubben and J. Morgenroth, "An odd couple: Loss-based congestion control and minimum RTT scheduling in MPTCP," in *Proceedings of the 44th IEEE Conference on Local Computer Networks (LCN)*, pp. 300–307, 2019.

[30] S. Kunniyur and R. Srikant, "End-to-end congestion control schemes: Utility functions, random losses and ECN marks," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 689–702, 2003.

[31] B. A. Movsichoff and C. M. Lagoa, "End-to-end optimal algorithms for integrated QoS, traffic engineering, and failure recovery," *IEEE/ACM Transactions on Networking*, vol. 15, no. 4, pp. 813–823, 2007.

[32] P. Hurley, J. Le Boudec, and P. Thiran, "A note on the fairness of additive increase and multiplicative decrease," in *Proceedings of the 16th International Teletraffic Congress*, IEEE, 1999.

[33] L. Ye, Z. Wang, H. Che, H. B. Chan, and C. M. Lagoa, "Utility function of TCP," *Computer Communications*, vol. 32, no. 5, pp. 800–805, 2009.

[34] F. Kelly, "Charging and rate control for elastic traffic," *European Transactions on Telecommunications*, vol. 8, no. 1, pp. 33–37, 1997.

[35] F. Kelly, A. Maulloo, and D. Tan, "Rate control in communication networks: Shadow prices, proportional fairness and stability," *Journal of the Operations Research Society*, vol. 49, no. 3, pp. 237–252, 1998.

[36] "Multipath TCP in the linux kernel." https://www.multipath-tcp.org. Accessed on Apr., 2021.

[37] W. Wei, K. Xue, J. Han, and P. Hong, "Shared bottleneck-based congestion control and packet scheduling for multipath TCP," *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 653–666, 2020.