Performance Enhancement of Multipath TCP for Wireless Communications with Multiple Radio Interfaces

Pingping Dong, Jianxin Wang, Jiawei Huang, Haodong Wang, and Geyong Min

Abstract

Multipath TCP (MPTCP) allows a TCP connection to operate across multiple paths simultaneously and becomes highly attractive to support the emerging mobile devices with various radio interfaces and to improve resource utilization as well as connection robustness. The existing multipath congestion control algorithms, however, are mainly loss-based and prefer the paths with lower drop rates, leading to severe performance degradation in wireless communication systems where random packet losses occur frequently. To address this challenge, this paper proposes a new mVeno algorithm, which makes full use of the congestion information of all the subflows belonging to a TCP connection in order to adaptively adjust the transmission rate of each subflow. Specifically, mVeno modifies the additive increase phase of Veno so as to effectively couple all subflows by dynamically varying the congestion window increment based on the receiving ACKs. The weighted parameter of each subflow for tuning the congestion

P. Dong, J. Wang and J. Huang are with the School of Information Science and Engineering, Central South University, Changsha, China, 410083. E-mail: {ppdong, jxwang, jiaweihuang}@csu.edu.cn

H. Wang is with the Department of Computer and Information Science, Cleveland, State University, Cleveland, OH, 44115, USA. E-mail: hwang@cis.csuohio.edu

G. Min is with the Department of Mathematics and Computer Science, University of Exeter, Exeter, EX4 4QF, United Kingdom. E-mail: g.min@exeter.ac.uk

window is determined by distinguishing packet losses caused by random error of wireless links or by network congestion. To validate the effectiveness of the proposed mVeno algorithm, we not only give the theoretical proofs, but also implement it in a Linux server and conduct extensive experiments both in testbed and in real WAN. The performance results demonstrate that mVeno increases the throughput significantly, achieves load balancing and can keep the fairness with regular TCP compared to the existing schemes.

Index Terms

Multipath TCP, congestion control, wireless networks, packet loss differentiation, throughput.

I. Introduction

Nowadays, portable devices, including smart-phones, tablets, and laptops, with multiple wireless interfaces such as WiFi and 4G/LTE, are becoming more and more popular [1]–[3]. The legacy transport protocols, such as Transmission Control Protocol (TCP), traverse one route using only one access interface and thus shields the user from the multipath features of wireless networks [4]. To enhance the resource usage while staying robust against link/network failures, Multipath TCP (MPTCP) [5], has been proposed recently by the Internet Engineering Task Force (IETF) working group. MPTCP allows a single data stream to be split into multiple subflows across multiple paths for improving throughput, robustness, and network resource utilization. MPTCP is a TCP extension and is envisioned to coexist with single-path TCP flows such that the applications over MPTCP can benefit from the use of available capacity on multiple paths without degrading the performance of plain TCP applications. MPTCP has received growing interests from both the academic and industrial communities. For example, MPTCP has already been implemented in the Siri applications of iPhones and iPads.

For the design of MPTCP, the following three constraints must be satisfied [6], [7]: (i) Performance enhancement: MPTCP should at least perform as a single-path TCP running on the best path. (ii) Bottleneck fairness: MPTCP should be TCP friendly, i.e., it should fairly share the

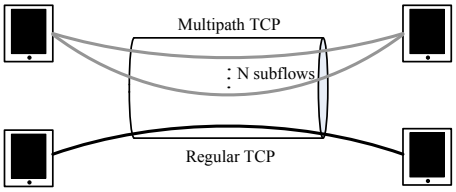


Fig. 1. A regular TCP and an MPTCP compete shared bottleneck.

bandwidth with the existing single-path TCP on a bottleneck link. (iii) Load balancing: MPTCP should move the traffic off its most congested paths as much as possible. If MPTCP runs the independent congestion control on each subflow, the fairness on the bottleneck links cannot be guaranteed [6], [8]. Take the network scenario shown in Fig. I as an example, where two clients share the same bottleneck link. If the MPTCP-enabled client uses two subflows, it will obtain two-thirds bandwidth of the bottleneck, which is unfair because it would have obtained only half of the shared bottleneck if this client was with regular TCP. To build MPTCP protocols compatible with the regular TCP, the existing MPTCP congestion control protocols couple the additive increase function of all the subflows belonging to a multipath flow together by assigning a weighted parameter of the regular TCP on each path. The parameter controls the increase rate of the congestion window on receiving new ACKs. The major challenge is how to calculate the weighted parameter to achieve performance enhancement, bottleneck fairness as well as load balancing simultaneously [8]–[10]. So far, several MPTCP congestion control algorithms have been proposed [7], [8], [11], [12]. However, most of these MPTCP congestion algorithms, such as Linked increase algorithm (LIA) [7], which is the currently adopted MPTCP congestion algorithms, use packet loss as the indicator to undertake load balancing and congestion control. This may cause severe performance degradation over wireless networks for two key reasons.

Firstly, these MPTCP algorithms tend to push traffic on the paths with the lower drop rates, leading to performance degradation when the path with the lower drop probability is not the

less congested path. This is very common in wireless networks where packet losses are more likely due to the transmission errors rather than the network congestion. Paasch [13] has also validated this problem, conducted experiments based on Linux testbed, and revealed that both COUPLED and OLIA algorithms cannot gain performance enhancement when random error exists. However, they did not give a solution to cope with the problem.

Secondly, most of the prior MPTCP congestion algorithms are loss based, assuming a relatively reliable underlying network where packet losses occur mainly because of congestion. However, wireless networks also suffer from significant packet loss due to bit errors and handoffs [14], [15]. The existing MPTCP algorithms respond to all losses by invoking congestion control and avoidance algorithms, causing performance degradation.

To address the above issues and improve the performance of MPTCP over wireless networks, this paper makes the following contributions:

1) An enhanced MPTCP congestion control algorithm, namely mVeno, is proposed for concurrent multipath transfer. To effectively couple the additive increase phase of each subflow, mVeno assigns various weights for different subflows and adaptively adjusts them to control the increment of sending rates of a subflow upon receiving new ACKs, balances the loads between the subflows, while ensuring fair bandwidth allocation to regular TCP at the shared bottleneck.
2) A key challenge in the proposed mVeno algorithm is to identify the weights suitable for different subflows of a TCP connection. To this end, the fluid flow model and the utility function of mVeno are designed in order to obtain the relationship between the sending rate and the end-to-end packet loss rate at the network equilibrium state. Theoretical analysis is then conducted to investigate the suitable weights and achieve the following objectives: (a) To balance the traffic load distribution among the subflows of a TCP connection by distinguishing packet losses between random error of wireless links and network congestion in order to achieve the equivalent loss rate caused by congestion of every subflow; (b)

To improve throughput and preserving fairness by ensuring that the total throughput of an mVeno flow is equivalent to that of a Veno flow on its best path.

3) We implement mVeno both in a testbed and in real WAN and conduct extensive experiments to validate its effectiveness. The results obtained from thorough comparison with the existing MPTCP algorithms, namely OLIA [11], wVegas [8], Balia [12] and MPVeno [16] demonstrate that mVeno can effectively utilize the network resource and significantly increase the network throughput without loss of fairness.

The rest of the paper is organized as follows. Section II discusses the related work. Section III presents the analysis model of TCP mVeno, investigates the weighted parameter and presents the mVeno algorithm. Section IV conducts in-depth analysis of the performance results obtained from real-word experiments. Finally, Section V concludes the paper.

II. Related Work and Existing Problems

Many studies have been conducted on transmitting data through multiple paths simultaneously, like Parallel TCP (pTCP) [17], Concurrent Transfer Multipath (CMT) over stream control transmission protocol (SCTP) [18], Wireless multipath SCTP (WiMP-SCTP) [19], Westwood SCTP [20], and Multiple paths TCP (mTCP) [21]. However, most of these schemes are uncoupled because each subflow performs congestion control independently of other subflows. As described in Section I, the uncoupled schemes become unfair to competing with single-path traffic in the general case.

To solve this problem, MPTCP has been proposed by modifying the additive increase during the congestion avoidance phase, where each subflow increases its congestion window according to the network condition of all the subflows belonging to a TCP connection. So far, several congestion control mechanisms have been developed for MPTCP. For example, the congestion window of each subflow in the fully coupled congestion algorithm (COUPLED) [7] was made to increase or decrease by considering the status of all subflows. The COUPLED algorithm can

make a multipath flow shift all its traffic onto the least-congested path. However, when the path with the highest loss rate is not the most congested link, COUPLED cannot effectively utilize the network resource, leading to performance degradation. To solve these problems, the Linked Increases algorithm (MPTCP-LIA) [6], [7] was proposed to satisfy the MPTCP constraints. MPTCP-LIA can guarantee that the total throughput of an MPTCP flow is at leat equivalent to that of a TCP flow on its best path. However, it has been proven in [11] that MPTCP-LIA is non-Pareto optimality, where MPTCP-LIA users could be aggressive towards single-path users. The MPTCP-OLIA algorithm [11] was proposed to solve the problem by improving the mechanism of the congestion window increasement. Meanwhile, Peng, Walid and Low [22] proved that the MPTCP-LIA algorithm is still unfair to single-path TCP and then proposed a new revision of MPTCP algorithm based on a fluid model. Balia algirithm [12] has been proposed recently, aiming at striking a good balance among TCP-friendliness, responsiveness, and window oscillation. However, as described in Section I, these loss-based congestion control schemes interpret a loss as congestion. They move traffic away from lossy subflows and halve the congestion window. However, this is not suitable for wireless communications where packet losses are often caused by random error rather than by network congestion [23], leading to severe performance degradation over wireless links.

More recently, several MPTCP congestion control algorithms have been proposed for wireless communications [9], [16]. These algorithms are the extended version of wireless TCP for MPTCP, like TCP Westwood [24] and TCP Veno [25]. Specifically, MPTCPW [9] is an extension of Westwood TCP for multipath transfer, based on the analysis model of Westwood TCP as well as the constraint that MPTCP performance should be equivalent to that of a regular TCP flow on its best path. MPVeno [16] was derived based on TCP Veno where the increase rule adopts MPTCP-OLIA design technique. Although these MPTCP mechanisms based on wireless TCP do not simply halve the congestion window when packet loss events incur, they can not effectively select the paths in wireless networks with random packet losses as they prefer the paths with

the lower packet drop rate and the selected path with the lower drop rate may be in a more congested state. In addition, the authors in [8] developed a delay-based MPTCP congestion control algorithm named wVegas, which adopts the packet queuing delay as congestion signals to achieve fine-grained congestion balance. However, it is unable to effectively aggregate the bandwidth of the subflows.

III. The proposed mVeno Algorithm

The goal of the mVeno algorithm is to couple all the subflows effectively in order to improve the overall performance over wireless networks by assigning various weighted parameters for different subflows to control the sending rate of each subflow. In this section, we first describe the fluid model of mVeno. Then, the derivation of the weighted parameter of each subflow based on the fluid-flow model is presented. Finally, the mVeno algorithm is designed.

A. Fluid-flow Model

For the multipath transfer, we consider a network shared by a set $S = \{1, ..., s\}$ of flows. Each flow $s \in S$ consists of a set of subflows R_s, each of which may take a different route. Every subflow $r \in R_s$ maintains its own congestion window $w_{s,r}$ and transmission rate $x_{s,r}$. An MPTCP sender stripes packets across these subflows as the space in the subflow windows become available. Let $y_s := \sum_{r \in R_s} x_{s,r}$ be the total rate of flow s.

mVeno is based on TCP Veno for multipath transfer. Specially, mVeno modifies the additive increase during the congestion avoidance phase by assigning different weights $\delta_{s,r}$ for different paths r to effectively couple the subflows.

TCP Veno is based on the idea of congestion monitoring scheme in TCP Vegas and integrates it into Reno's congestion avoidance phase. Briefly speaking, Veno calculates the backlog N at the queue and uses it as an indicator of whether the network is congestive or not. If $N \geq \beta$, the network is in congestive state. Veno increases the congestion window $w(t)$ by $1/w(t)$ on every

other positive acknowledgment and decreases it by half on each packet loss event. Otherwise, if $N < \beta$, it is in the non-congestive state. The congestion window $w(t)$ is increased by $1/w(t)$ on every positive acknowledgment and decreased by one-fifth on each packet loss event [25]. The backlog N is given by Eq. (1).

$$N = (\frac{cwnd}{baseRTT} - \frac{cwnd}{RTT}) * baseRTT \tag{1}$$

where $cwnd$ is the congestion window size, RTT is the average RTT in the last round and $baseRTT$ is the minimal RTT that has been measured so far.

mVeno refines Veno's additive increase algorithms through the distribution of the weighted parameter $\delta_{s,r}$ for each subflow. The algorithm is shown in Algorithm 1 in Section III-B and the implementation is given in Section IV-A. Specifically, upon receiving new ACKs, the congestion window increment is $\delta_{s,r}$ rather than 1. In other words, on each subflow, mVeno behaves as follows: If $N \geq \beta$, mVeno flow s increases the congestion window $w_{s,r}(t)$ on subflow r by $\delta_{s,r}/w_{s,r}(t)$ on every other positive acknowledgment and decreases it by half on each packet loss event. Otherwise, if $N < \beta$, the congestion window $w_{s,r}(t)$ is increased by $\delta_{s,r}/w_{s,r}(t)$ on every positive acknowledgment and is decreased by one-fifth on each packet loss event. Let $p_{s,r}(t)$ be the packet drop probability. Putting the increase and decrease rate together, on average, the source receives $w_{s,r}(t)(1 - p_{s,r}(t))$ positive acknowledgments per time unit. It receives, on average, $w_{s,r}(t)p_{s,r}(t)$ negative acknowledgments per time unit [26]. Neglecting the delayed increase behavior, as the most enhancement of Veno is contributed by the MD (Multiplicative Decrease) phases at random losses rather than the delayed increase of congestion window at the additive increase phases [27], we get the fluid models of mVeno and Veno corresponding to the AIMD mechanisms, in respect of continuous time t as described in Eq. (2) and Eq. (3), respectively.

$$\frac{dw_{s,r}(t)}{dt} = \frac{w_{s,r}(t)}{T_{s,r}}(\frac{(1-p_{s,r}(t))\delta_{s,r}}{w_{s,r}(t)} - E_{s,r}[\eta_i]p_{s,r}(t)w_{s,r}(t)) \tag{2}$$

$$\frac{dw(t)}{dt} = \frac{w(t)}{T}(\frac{1-p(t)}{w(t)} - E[\eta_i]p(t)w(t)) \tag{3}$$

where $E_{s,r}[\eta_i]$ and $E[\eta_i]$ are the average decrease factor, $w(t)$ and $w_{s,r}(t)$ are the congestion window size at time t, $p(t)$ and $p_{s,r}(t)$ are the packet drop rate, T and $T_{s,r}$ are the equilibrium round-trip time which can be assumed as a constant [26]. The average decrease factor is calculated as (taking $E[\eta_i]$ as an example) $E[\eta_i] = \frac{1}{5} * P(N < \beta) + \frac{1}{2} * P(N \geq \beta)$, where $P(N < \beta)$ is the probability that N is less than β and can be expressed as [27]:

$$P(N < \beta) = P(cwnd < \frac{\beta * RTT}{RTT - baseRTT}) = \min(1, \frac{\beta * RTT}{(RTT - baseRTT) * W_{\max}}) \tag{4}$$

Thus,

$$E[\eta_i] = \max(\frac{1}{n}, \frac{1}{2} - (\frac{1}{2} - \frac{1}{n}) * \frac{\beta * RTT}{(RTT - baseRTT) * W\max}) \tag{5}$$

where $Wmax$ is the maximum congestion window size during the transmission.

B. mVeno Algorithm

In this subsection, based on the fluid model of mVeno, we first employ the network dual utility model to couple subflows to achieve load balancing between paths.

As the fixed point of the fluid-flow model defines an implicit relation between the equilibrium rate and the end-to-end congestion measure [26]. Thus, for TCP Veno, we can obtain the relationship between the equilibrium rate x (defined by $x = \frac{w}{T}$) and packet loss rate p as described in Eq. (6), by setting Eq. (3) to zero.

$$p = \frac{1}{1 + E[\eta_i]x^2T^2} =: f(x) \tag{6}$$

The utility function is calculated as [26]

$$U(x) = \int f(x)dx, x \geq 0 \tag{7}$$

Thus, we can obtain the utility function of Veno:

$$U(x) = \frac{1}{T\sqrt{E[\eta_i]}}\tan^{-1}(T\sqrt{E[\eta_i]})x \tag{8}$$

Clearly, $U(x_s)$ is increasing, strictly concave and twice continuously differentiable in the nonnegative domain.

Based on the utility function of Veno for single-path user in Eq. (8), we construct the utility function of mVeno as follows [8], [28]:

$$U_s(y_s) = \frac{1}{T_s\sqrt{E_s[\eta_i]}}\tan^{-1}(T_s\sqrt{E_s[\eta_i]})y_s \tag{9}$$

To satisfy the fairness constraints as validated in Section III-C, T_s is the maximum value of the average RTT of all subflows belonging the multipath TCP flow s. $E_s[\eta_i]$ is the minimum value of decrease factor of the multipath TCP flow s.

The derivative of the above utility function is:

$$U'(y_s) = \frac{1}{1 + E_s[\eta_i]T_s^2y_s^2} \tag{10}$$

$U'(y_s)$ can be interpreted as the expected congestion extent [26], as though flow s transferred data at the rate y_s along a virtual single path whose RTT is T_s and whose capacity is the sum of all the paths used by flow s.

Meanwhile, from the subflow's perspective, at the equilibrium point, Eq. (2) shows the relationship between the sending rate $x_{s,r}$ and the packet loss probability $p_{s,r}$ for each subflow $r \in R_s$:

$$\frac{(1 - p_{s,r})\delta_{s,r}}{x_{s,r}T_{s,r}} = E_{s,r}[\eta_i]p_{s,r}x_{s,r}T_{s,r} \tag{11}$$

It has been proven that effectively achieving load balancing between subflows means that every subflow should strive to equalize the extent of congestion that it perceives on all its available paths [8]. Traditionally, the packet dropping probability $p_{s,r}/U'(y_s)$ is assumed to quantify the congestion extent of networks. However, in lossy wireless networks, the packet loss is often induced by noise, link error, or reasons other than network congestion. Thus, we propose that the traffic distribution between subflows should satisfy the constraint that the packet dropping probability caused by congestion on each subflow should be equivalent with distinguishing packet losses between random error of wireless links and network congestion in wireless networks. Based on the proposal, we can obtain Eq. (12) where the loss rate caused by congestion on any subflow equals the expected congestion extent.

$$P_{s,r}(N > \beta) * p_{s,r} = P_s(N > \beta) * \mathrm{U}'(\mathrm{y}_s) \tag{12}$$

Comboning Eqs. (10), (11), (12), we can derive that:

$$\delta_{s,r} = \frac{P_s(N > \beta) E_{s,r}[\eta_i] w_{s,r}^2}{P_{s,r}(N > \beta)(1 + E_s[\eta_i] T_s^2 y_s^2) - P_s(N > \beta)} \tag{13}$$

According to the calculated weighted parameter $\delta_{s,r}$ for each subflow r, we design the algorithm of mVeno as shown in Algorithm 1.

In this Algorithm, β is assigned as the default value of 3 [29], [30] that is consistent with the current Linux implementation, and the initial weighted parameter for each subflow is set $1/M$ where M is the number of subflows. The weighted parameter $\delta_{s,r}$ is adjusted once per RTT round as shown in the $Linkpara\_Adjust()$ process (Lines 19-21) according to Eq. (13). Meanwhile, as shown in Eq. (13), the calculation of $\delta_{s,r}$ involves several variables, namely, $baseRTT_{s,r}$, $diff_{s,r}$, $W_{s,r}^{\max}$, $T_{s,r}$ and $x_{s,r}$ for subflow r, $baseRTT$, $W_s^{\max}$, T_s and y_s for MPTCP flow s. These parameters are calculated and updated based on the measurement during the last round in order to quickly respond to the changes of network conditions as depicted in Lines 6-13. The

Algorithm 1: The mVeno Algorithm

Initialization at $t = 0$

  for *each* $r \in R_s$ **do**

    $\delta_{s,r} = 1/M$;

    $\beta_{s,r} = 3$;

Upon receiving a new ack **on subflow** r**:**

  /* perform per-round-operations */

  $T_{s,r} \leftarrow \sum_r sampledRTT_{s,r}/sampledNum_{s,r}$;

  $baseRTT_{s,r} \leftarrow min\{sampledRTT_{s,r}\}$;

  $baseRTT \leftarrow min\{baseRTT_{s,r}\}$;

  $diff_{s,r} \leftarrow cwnd_{s,r} * (T_{s,r} - baseRTT_{s,r})/T_{s,r}$;

  $x_{s,r} \leftarrow cwnd_{s,r}/T_{s,r}$;

  $y_s = \sum_{r \in R_s} x_{s,r}$;

  $T_s = \frac{\sum_{r \in R_s} \mathrm{sampledRTT}_{s,r}}{\sum_{r \in R_s} \mathrm{sampledNum}_{s,r}}$;

  Calculate $E_s[\eta_i]$ and $E_{s,r}[\eta_i]$ according to $Eq.$ (5);

  Linkpara_Adjust();

  /* perform per-ack-operations */

  if $diff_{s,r} < \beta$ **then**

    $w_{s,r} \leftarrow w_{s,r} + \delta_{s,r}/w_{s,r}$;

  else

    $w_{s,r} \leftarrow w_{s,r} + \delta_{s,r}/(m * w_{s,r})$;

Linkpara_Adjust():

  for *each* $r \in R_s$ **do**

    $\delta_{s,r} = \frac{P_s(N>\beta)E_{s,r}[\eta_i]T_{s,r}^2 x_{s,r}^2}{P_{s,r}(N>\beta)(1+E_s[\eta_i]T_s^2 y_s^2) - P_s(N>\beta)}$;

instant rate $x_{s,r}$ of subflow r is updated by dividing $cwnd_{s,r}$ by $T_{s,r}$. $cwnd_{s,r}$ is the average value of the congestion windows size of the sampled packets on subflow r. $T_{s,r}$ is the average RTT measured on subflow r of the corresponding packets in the current round. T_s is the average RTT of all the sampled packets from all the subflows and y_s is the sum of the rate $x_{s,r}$ for each subflow r belonging to s. $baseRTT_{s,r}$ and $W_{s,r}^{\max}$ are the minimum of all RTT measurements and the maximum of all congestion window size seen on the path r. $baseRTT$ and $W_s^{\max}$ are the corresponding values of all the subflows. Based on these parameters, $E_s[\eta_i]$ and $E_{s,r}[\eta_i]$ are calculated according to Eq. (5) and finally the weighted parameter $\delta_{s,r}$ can be obtained.

According to the updated $\delta_{s,r}$, mVeno modifies TCP Veno's additive increase algorithm of the steady-state congestion avoidance phase so as to achieve throughput enhancement, load balancing and bottleneck fairness simultaneously.

C. Algorithm Analysis

In this subsection, we discuss the three properties of mVeno algorithm, namely, throughput enhancement, load balancing and bottleneck fairness.

(1) *Throughput enhancement: mVeno algorithm can achieve throughput enhancement such that the congestion window size for a path r of mVeno is larger than that of LIA, under the same network scenarios.*

When analyzing this property, we first derive the expression of the congestion window of each subflow. Then a comparison of the window size of the two algorithms is shown in Fig. 2.

Based on Eq. (2), at the equilibrium point, we can obtain a simple loss-throughput formula for mVeno as shown in Eq. (14) where $1 - p_{s,r} \approx 1$ is adopted as that in [6].

$$w_{s,r} \approx \sqrt{\frac{\delta_{s,r}}{E_{s,r}[\eta_i]p_{s,r}}} \tag{14}$$

As analyzed in [27], the congestion window size at the equilibrium state of TCP Veno is:

$$\hat{w}_{s,r} = \sqrt{\frac{1}{E_{s,r}[\eta_i]\hat{p}_{s,r}}} \tag{15}$$

Following the similar derivation process of the congestion window size for LIA in Ref. [11], by substituting Eq. (15) into Eq. (13) and then by substituting Eq. (13) into Eq. (14), we obtain Eq. (16).

$$w_{s,r} \approx \frac{1}{p_{s,r}} * \frac{1}{\sqrt{E_{s,r}[\eta_i]}} * \sqrt{\frac{P_s(N > \beta)}{P_{s,r}(N > \beta)(1 + E_s[\eta_i](\sum_r \sqrt{1/E_{s,r}[\eta_i]p_{s,r}})^2) - P_s(N > \beta)}} \tag{16}$$

As analyzed in [11], the congestion window size for flow s on a path r of LIA is given by:

$$w_{s,r} = \frac{1}{p_{s,r}} * \frac{\max_r \sqrt{2/p_{s,r}} \Big/ rtt_{s,r}}{\sum_r 1/(rtt_{s,r}p_{s,r})} \tag{17}$$

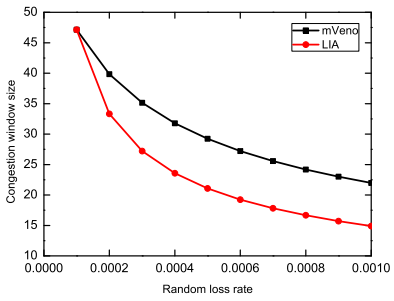


Fig. 2. The congestion window size versus the random loss rates.

Fig. 2 depicts the window sizes of mVeno versus LIA according to Eq. (16) and Eq. (17), respectively. In this figure, we consider an MPTCP flow consisting of two subflows, which have the same network configuration. The congestion loss rate is kept at 0.01% and the random loss rate varies from 0% to 0.05%. The loss rate of subflow r is calculated as $p_{s,r} = 1-(1-x_{s,r})(1-y_{s,r})$ where $x_{s,r}$ is the congestion loss rate and $y_{s,r}$ is the corresponding random loss rate. In the mVeno algorithm, the parameter $P_{s,r}(N > \beta)$ is calculated as $x_{s,r}/p_{s,r}$, which is the estimation

of the probability that the loss is caused by congestion [25]. As shown in the figure, due to the packet loss differentiating mechanism, mVeno can achieve a larger congestion window size.

(2) *Load balancing: mVeno algorithm can achieve load balancing such that every subflow has almost the same congestion extent.*

To investigate the load balancing property, we first present the theoretical proof and then give a network instance.

Proof: Since $\delta_{s,r}$ controls the bandwidth obtained by flow s on path r, we analyze whether it can effectively achieve congestion balance or not in this theorem. As described above, if $P_{s,r}(N > \beta) * p_{s,r} > P_s(N > \beta) * \mathrm{U}'(\mathrm{y}_s), \delta_{s,r}(t)$ should be decreased so as to partially offload the traffic. That is, $\delta_{s,r}(t+1)$ should be smaller than $\delta_{s,r}(t)$.

From Eq. (11), we can derive that

$$p_{s,r} = \frac{\delta_{s,r}}{\delta_{s,r} + E_{s,r}[\eta_i] w_{s,r}^2(t)} \tag{18}$$

Substituting Eq. (18) and Eq. (10) into $P_{s,r}(N > \beta) * q_{s,r} > P_s(N > \beta) * \mathrm{U}'(\mathrm{y}_s)$ yields

$\frac{\delta_{s,r} P_{s,r}(N>\beta)}{\delta_{s,r}+E_{s,r}[\eta_i] w_{s,r}^2(t)} > \frac{P_s(N>\beta)}{1+E_s[\eta_i] T_s^2 y_s^2}$

Thus $\delta_{s,r}(t) > \frac{P_s(N>\beta) E_{s,r}[\eta_i] w_{s,r}^2(t)}{P_{s,r}(N>\beta)(1+E_s[\eta_i] T_s^2 y_s^2) - P_s(N>\beta)} = \delta_{s,r}(t+1)$ holds. □

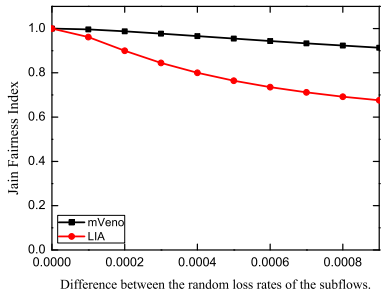


Fig. 3. Matlab validation of the load balancing property of the mVeno algorithm where the loss rate is fixed at 0.01%, the random loss rate of one subflow is fixed at 0.01%, while that of the other subflow changes from 0.01% to 0.1% with a step of 0.01%.

To present the property more clearly, we construct a scenario that the subflows have the same congestion loss rate and different random loss rates. In such a network configuration, the throughput difference between each subflow with mVeno should be less than that of LIA. The Jain Fairness Index metric is adopted to represent the difference between subflows. The results are shown in Fig. 3. As described in this figure, with the increment of the difference between the random loss rates of the subflows, the Jain Fairness Index decreases. Benefitting from the packet loss distinguishing mechanism when choosing paths, mVeno can utilize the path with high random loss rate more efficiently and performs better, which is consistent with the experimental results shown in Section IV.

(3) *Bottleneck fairness: mVeno algorithm is TCP friendly when the single-path TCP exists on a bottleneck link.*

To validate the bottleneck fairness of mVeno when competing with regular TCP, we prove that the window increment on each subflow flow r is less than a single-path TCP flow would.

Proof: As analyzed in the previous literatures, an MPTCP flow can achieve fairness when the congestion window increment on each subflow flow r is less than the increment of a single-path TCP flow on the same link. In other words, $\delta_{s,r}$ should be less than 1. As a result,

$$1 - \frac{P_s(N > \beta)E_{s,r}[\eta_i]T_{s,r}^2 x_{s,r}^2}{P_{s,r}(N > \beta)(1 + E_s[\eta_i]T_s^2 y_s^2) - P_s(N > \beta)} \geq 0 \tag{19}$$

$$\begin{aligned} &\frac{P_{s,r}(N>\beta) - P_s(N>\beta) + 0.3P_{s,r}(N>\beta)P_s(N>\beta)(T_s^2 y_s^2 - T_{s,r}^2 x_{s,r}^2)}{P_{s,r}(N>\beta)(1+(0.3P_s(N>\beta)+0.2)T_s^2 y_s^2) - P_s(N>\beta)} \\ &+ \frac{0.2P_{s,r}(N>\beta)T_s^2 y_s^2 - 0.2P_s(N>\beta)T_{s,r}^2 x_{s,r}^2}{P_{s,r}(N>\beta)(1+(0.3P_s(N>\beta)+0.2)T_s^2 y_s^2) - P_s(N>\beta)} \geq 0 \end{aligned} \tag{20}$$

Substituting $E[\eta_i] = \frac{1}{5} * (1 - P(N \geq \beta)) + \frac{1}{2} * P(N \geq \beta)$ into Eq. (19). We can obtain Eq. (20). It is clear that when $P_s(N > \beta)$ adopts the minimum and T_s adopts the maximum value, Eq. (20) always holds. This ensures that the mVeno flow can take no more capacity on either path than a single-path TCP flow would.

□

IV. PERFORMANCE EVALUATION

In this section, we validate the proposed mVeno algorithm by performing the extensive experiments on our testbed as well as in a real WAN network and comparing its performance results with other four existing algorithms: wVegas [8], Balia [12], OLIA [11] and MPVeno [16]. We implemented mVeno and MPVeno in our Linux Server. The other three algorithms have been included in MultiPath TCP v0.9.

A. Implementation of mVeno

To conduct the experiments, we implemented mVeno and MPVeno as separate Linux models with kernel version 4.1 of Ubuntu 14.04 operation system. Similar to COUPLED or OLIA, we mainly focus on the congestion control dynamics of MPTCP. After the kernel version 2.6.13, Linux supports pluggable congestion control algorithms and the modular structure is implemented. Thus, to implement the mVeno algorithm, we first initialize the $tcp\_congestion\_ops$ structure, which is a collection of the description for different pluggable congestion control algorithms, and then make a call to the $tcp\_register\_congestion\_control$ to register the new congestion control mechanism. The congestion control mechanisms described in Algorithm 1 are implemented in the $mptcp\_mveno\_cong\_avoid()$ function. In mVeno, the Veno's additive increase phase is refined with the weighted parameter $\delta_{s,r}$. As the Linux implementation of TCP Veno is well known, we mainly focus on the implementation of calculation of the coordination parameter $\delta_{s,r}$ as described in Eq. (13).

mVeno congestion control couples different subflows with different $\delta_{s,r}$ for the increase phase. As described in Section III-B, the parameters, like $T_{s,r}, x_{s,r}, E_{s,r}, T_s, y_s$ and E_s are needed for the calculation of $\delta_{s,r}$. mVeno uses a RTT round as the control interval on each subflow so as to quickly respond to the changes in the extent of network congestion.

As the congestion window and the RTT are already defined in the Linux kernel as $tp \rightarrow snd\_cwnd$ and rtt_us, respectively, and can be available to a regular TCP user, the average

RTT $T_{s,r}$ can be calculated by dividing the sum of rtt_us by the count of the sampled packets of subflow r. Consequently, the sending rate $x_{s,r}$ can be updated by dividing the average value of $tp \rightarrow snd\_cwnd$ by $T_{s,r}$. Similarly, other parameters for subflow r can be also obtained step by step as analyzed in Section III-B. Besides, to calculate the parameters of the MPTCP flow, such as T_s and y_s, the function $mptcp\_for\_each\_sk$ needs to be used to obtain variable values, such as $x_{x,r}$ and $T_{s,r}$, from each subflow.

Based on these precalculated parameters, $\delta_{s,r}$ is adjusted at the end of each round as shown in Eq. (13). As the Linux kernel only handles fixed-point calculations, we need to scale the divisions, to reduce the error due to integer-underflow. In the next round, the recalculated parameters are assigned and updated to make AIMD congestion control on new ACKs or packet loss events.

B. Testbed Construction and Experimental methodology

The deployed experiment testbed consists of three file servers, two computers with WANem and three clients. Both the clients and the servers are running on Linux ubuntu 14.01 OS with kernel version 4.10. All of these computers are connected to a Gigabit switch. These components logically constitute three network topologies shown in Fig. 4, Fig. 10 and Fig. 13, by means of different routing configurations that are the typical topologies widely used in the existing studies [9], [12], [13], [16].

All nodes are running on Dell T1500, equipped with the Intel Xeon E5620 (2.4GHz/12M), 16 GB RAM, a 600 GB Hard Disk and multiple Gigabit Ethernet interfaces. In the testbed, each MPTCP flow is composed of two subflows as this is a common scenario (e.g., a client having two access networks like WiFi/4G) [13] and has been widely used in the previous studies [12], [13]. The network metrics such as loss rate, delay and bandwidth are controlled by WANem. A wide range of network environments are considered: the round-trip time is in the range from 20ms to 400ms, packet loss rate is changed from 0.1% to 5% and the bottleneck bandwidth also varies from 2Mbps to 20Mbps.

GNU Wget is used to generate TCP data traffic by retrieving binary documents through HTTP. The document sizes vary from 1MB to 8MB. All the experimental data is captured at the clients using tcpdump and then is analyzed with wireshark. The goodput is calculated as the ratio of the file size to the elapsed time for loading the whole document.

Our experiments are divided into four parts. First, we compare goodput enhancement of mVeno algorithm and prior algorithms. Second, we compare the congestion balance of each algorithm in a heterogeneous environment where two subflows have different network configurations. Third, we show the TCP friendliness of every algorithm when sharing the same bottleneck link with regular TCP flows. Finally, the performance in real WAN environment is presented.

C. Throughput

We study the goodput enhancement of the aforementioned five algorithms with the topology shown in Fig. 4. In the experiments, the number of the competing MPTCP flows is 10 and each MPTCP flow consists of two subflows. Both the subflows are set with the same network configuration. The results are depicted from Fig. 5 to Fig. 8. As shown in Fig. 5, the algorithms have a similar behavior with different file sizes. Considering that these algorithms mainly apply to the AIMD part of the congestion avoidance phase, we mainly focus on the steady-state goodput. Thus, in the following experiments, we take the 8MB file as an example.

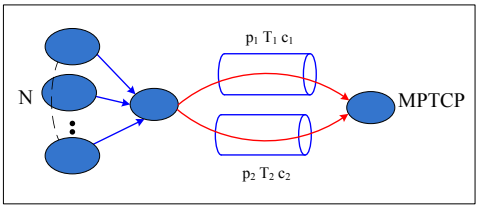


Fig. 4. Network for Linux-based experiments on goodput with 10 MPTCP flows. Each MPTCP flow maintains two subflows.

According to the experimental results, we can see that mVeno performs best, followed by MPVeno, wVegas, Balia and OLIA. The reason lies in that distinguishing the cause of the

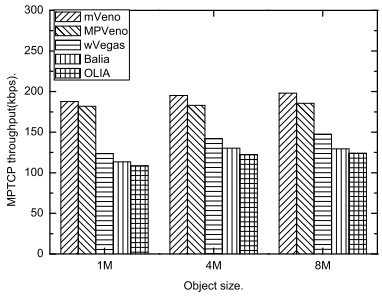


Fig. 5. Obtained goodput with varying binary document sizes. Parameters: T_1 = T_2 = 100ms, c_1 = c_2 = 8Mbps and p_1 = p_2 = 2%.

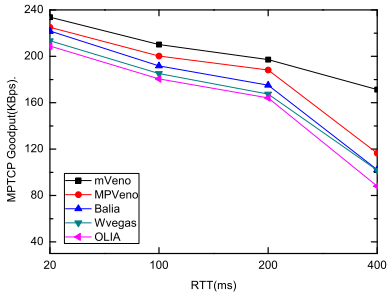


Fig. 6. Obtained goodput with varying RTTs. Parameters: c_1 = c_2 = 8Mb/s and p_1 = p_2 = 0.1%.

packet loss helps mVeno and MPVeno to avoid half reduction of congestion window caused by random link error. wVegas is a RTT based congestion control, which can also survive the random loss rates.

Moreover, mVeno also outperforms MPVeno. The reason is as follows. It is known that, Veno compares the backlog at the queue N with the threshold named β to decide whether the network is congestive or not. Specifically, if $N > \beta$, the network is said to have evolved into congestive state. Packet loss in the congestive state is considered as congestion loss and the source reduces its congestion window by half on every packet loss. Otherwise, packet loss is considered as random loss and the congestion widow is reduced by 1/5. In the MPVeno algorithm, the threshold β is

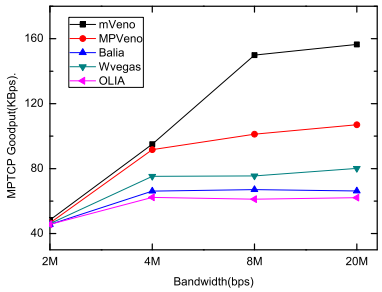


Fig. 7. Obtained goodput with varying bandwidth. Parameters: T_1 = T_2 = 100ms and p_1 = p_2 = 2%.

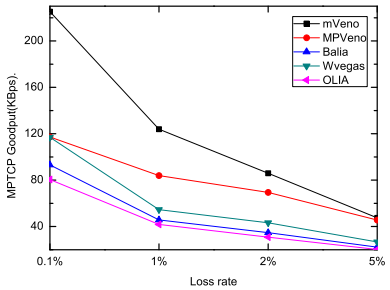


Fig. 8. Obtained goodput with varying loss rate. Parameters: T_1 = T_2 = 400ms and c_1 = c_2 = 20Mbps.

limited to $(x_{s,r}/y_s)*\beta$ to achieve bottleneck fairness, where $x_{s,r}$ is the rate of subflow r and y_s is the sum of the rates of all subflows. Obviously, $(x_{s,r}/y_s)*\beta$ is less than β. The smaller threshold value implies the increasing chance that MPVeno interprets a packet loss as a congestion. Thus, compared to mVeno, MPVeno is more likely to reduce the congestion window $w_{s,r}$ by $w_{s,r}/2$, causing the degraded performance.

As MPTCP allows subflows to change during the lifetime of the connection for various reasons, we then consider the environment where one subflow disappears at 60s. Specifically, 60s after the start of the test, we close one of the subflow by taking down the corresponding network interface card with the system command *ifconfig*. In the experiment, the loss rate, RTT and

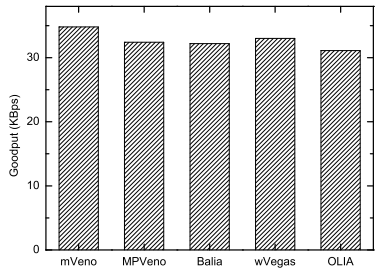


Fig. 9. The average goodput gain (KBps) of the MPTCP flows when one subflow disappears at 60s by configuring the network interface card down. The loss rate, RTT and bandwidth are set as 1%, 200ms and 2Mbps, respectively.

bandwidth are set as 1%, 200ms and 2Mbps, respectively. With this configuration, the MPTCP connection can last long enough (about 300s) for the test. The obtained goodput is shown in Fig. 9. The five algorithms reveal a similar behavior as the scenario of two subflows always existing, except that wVegas gains higher goodput than MPVeno does.

D. Load balancing

We adopt the topology shown in Fig. 10 to evaluate the effectiveness of mVeno in terms of load balancing. In the experiments, we consider one MPTCP flow and construct diverse configurations to investigate the capability of shifting traffic from the congested path of each algorithm, where the RTT or loss rate on the lightly loaded path is larger than that on the congested one. We use TCP Reno flows as the background traffic and vary the number of flows to generate different traffic loads on each subflow.

In the first set of experiments, the bandwidth and RTT of the two links are set to be the same value of 4Mb and 100ms, The packet loss rate of Link 1 and Link 2 is 2% and 1%, respectively. Besides, 11 TCP Reno flows also run on Link 2 to compete with subflow2, constituting a more congested link.

The aggregated data is depicted in Fig. 11. As shown in the figure, with the mVeno algorithm,

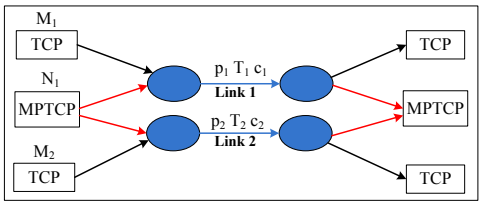


Fig. 10. Network for Linux-based experiments on load balancing.

subflow2 on the more congested Link 2 can obtain nearly the same goodput as that of regular TCP. Subflow1 of the mVeno flow on the less congested Link 1 achieves the highest goodput among the five MPTCP algorithms. Meanwhile, the goodput of regular TCP with mVeno algorithm is no less than that of other algorithms. These results again validate that mVeno can choose the paths effectively and can achieve a higher network utilization. It is worth noting that the obtained goodput of subflow2 is slightly larger than that of the regular TCP on the same link. The reason lies in that mVeno is based on Veno while the background regular TCP is Reno.

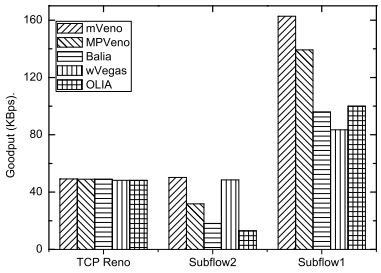


Fig. 11. The goodput gain (KBps) of regular TCP and each subflow with topology shown in Fig. 10. The RTT and bandwidth of the two links are 100ms and 4Mbps. The loss rates of Link 1 and Link 2 are 2% and 1%, respectively. 11 TCP Reno flows also run on Link 2.

Furthermore, we also carried out experiments that the loss rate is the same while the RTT and bandwidth are different. The link capacity and propagation delay of Links 1 and 2 are

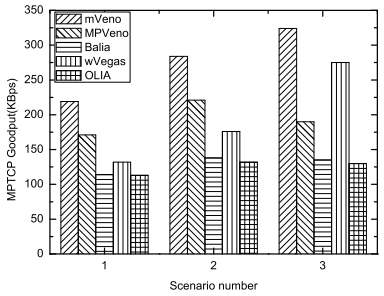


Fig. 12. Obtained goodput of MPTCP flows with the adverse network configurations. Scenario 1: with the link capacity and propagation delay 100ms and 4Mbps, the loss rate of Link 1 and Link 2 2% and 1%, respectively and 11 Reno flows on Link 2. Scenario 2: with the link capacity and propagation delay of Link 1 and 2 (6Mbps, 160ms) and (10Mbps, 40ms), loss rate 0.1%, 3 Reno flows on Link 1 and 8 Reno flows on Link 2. Scenario 3: with the link capacity and propagation delay of Link 1 and 2 (6Mbps, 160ms) and (10Mbps, 40ms), loss rate 1%, 3 Reno flows on Link 1 and 8 Reno flows on Link 2.

set (6Mbps, 160ms) and (10Mbps, 40ms), respectively. To generate heavy load on Link 1 and light load on Link 2, there are eight TCP Reno flows on Link 1, and three flows on Link 2. We considered two scenarios where the loss rate is 0.1% and 1%, respectively. The results are shown in Fig. 12. Consistent with the analysis of the results in Fig. 11, mVeno can achieve load balancing and obtain the higher goodput.

E. Fairness on Bottleneck Links

As described in the previous subsections, mVeno can achieve the higher network goodput. We will investigate in this subsection that the performance enhancement is not at the cost of fairness. The network topology for our fairness experiment is shown in Fig. 13, where 10 two-path MPTCP flows and 10 regular single-path TCP flows compete for the same bottleneck link.

As shown in Figs. 6-8, the goodput gain of mVeno increases with the increase of RTT and bandwidth. Thus, we can predict that if mVeno can achieve TCP fairness in this high bandwidth and large delay environment, mVeno should be always TCP friendly. Fig. 14 shows the Jain Fairness Index where the RTT is 400ms and bandwidth is 20Mbps. According to the figure, we

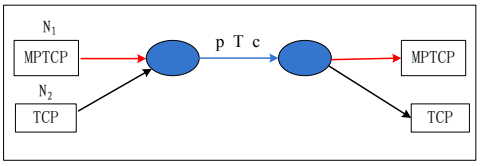


Fig. 13. Network for Linux-based experiments on fairness with 10 MPTCP flows and 10 TCP flows.

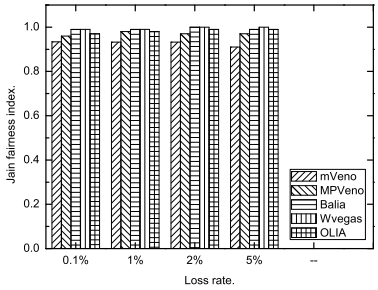


Fig. 14. Jain Fairness Index performance.Parameters: T = 400ms and c= 20Mbps.

can obtain that Balia and wVegas are the most friendly, followed by OLIA and MPVeno. In addition, mVeno performs worst. The results are unexpectedly. However, making more detailed analysis of aggregated data, we found another thing. Taking the loss rate 0.1% scenario as an example, the results shown in Fig. 15 reveal that the average TCP goodput of mVeno is no lower than that of other MPTCP algorithms while the MPTCP throughput of mVeno is much greater than other algorithms. This indicates that the goodput enhancement of mVeno is achieved by network utilization and not stealing the resource of regular TCP.

F. Real WAN

The real-world traces are collected from an operational WAN. The MPTCP server is located at Central South University, where the Internet is connect through Hunan Netcom, China. The client is installed with two wireless usb cards. One is connected to an AP and the other is connected

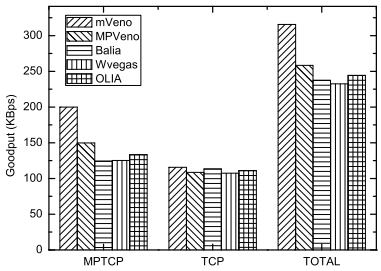


Fig. 15. The average goodput gain (KBps) of the MPTCP flows and regular TCP flows corresponding Jain Fairness Index shown in Fig. 14. The results show that algorithm mVeno has a lower Jain Fairness Index, the regular TCP throughput gain of mVeno is not lower than that of other MPTCP algorithms.

to a 4G router. The RTT and loss rate of WIFI link are 25ms and 3.5%-5%, while those of the 4G link are 20ms and 0%, respectively. As the network environment of the real-world is much more complex than the emulation environment. To keep a relative stable network environment, the experiments are executed at a fixed time of a day. The real world experiments last for one week as each data point is obtained by computing the average value of the results from at least ten rounds of executions.

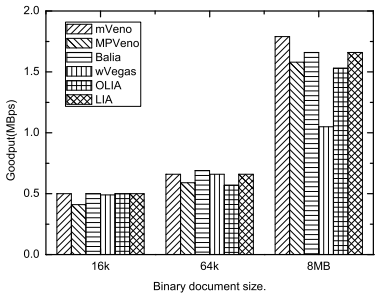


Fig. 16. Evolution in real traces.

The results are depicted in Fig. 16. When the file size is small, say 16KB or 64KB, the

obtained goodput of each algorithm differentiates slightly. For the 8MB file, the figure reveals that mVeno performs best, LIA and Balia come the second, followed by OLIA, MPVeno and wVegas.

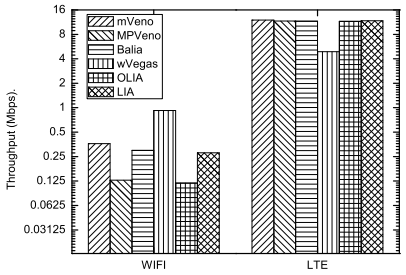


Fig. 17. Obtained throughput(Mbps) of both WIFI and 4G link in real WLAN traces. It is to note that we use *throughput* instead of *goodput* in the figure as the throughput are analyzed with the *Statistics–Summary* command of wireshark without filtering the retransmissions.

According to the analysis data shown in Fig. 17, except for wVegas, mVeno achieves the higher throughput on the WIFI link because the packet loss differentiation mechanism is applied both in the congestion avoid phase and in the load balancing. Balia and LIA also show a good performance in this experiment. As OLIA and MPVeno adopt the same weights parameters, they have a similar behavior, where the Veno based MPVeno performs better owing to the gains of the Veno algorithm in wireless networks. The poor performance of wVegas in this experiment mainly lies in that wVegas takes load balance based on RTT. As both the WIFI link and the LTE link have the similar RTT, compared to other algorithms, wVegas shifts too much traffic on the poor WIFI link and thus obtains the lower throughput.

V. Conclusions

In the paper, we have proposed an enhanced MPTCP congestion control algorithm over wireless networks. mVeno refines Veno's additive increase algorithm by assigning different

weighted parameters for different paths to couple all the subflows of a TCP connection. Based on the AIMD algorithm of mVeno in the steady-state congestion avoidance phase, we have developed the fluid model of mVeno, and then derived the weighted parameter with the goal of satisfying the MPTCP constraints, namely, performance enhancement, load balancing and bottleneck fairness simultaneously. We analyzed the properties of mVeno theoretically and implemented mVeno in the Linux server. The results obtained from extensive testbed experiments as well real WAN environment demonstrate that mVeno outperforms the existing schemes. Future investigations will focus on exploring an improved MPTCP algorithm for HTTP like short flows and a balanced trade-off between power consumption and use of multiple paths for mobile devices.

REFERENCES

[1] A. Makhlouf and M. Hamdi, "Practical rate adaptation for very high throughput wlans," *IEEE Transactions on Wireless Communications*, vol. 12, no. 2, pp. 908–916, February 2013.

[2] B. Makki and T. Eriksson, "On the performance of mimo-arq systems with channel state information at the receiver," *IEEE Transactions on Communications*, vol. 62, no. 5, pp. 1588–1603, May 2014.

[3] M. Ismail, A. Gamage, W. Zhuang, X. Shen, E. Serpedin, and K. Qaraqe, "Uplink decentralized joint bandwidth and power allocation for energy-efficient operation in a heterogeneous wireless medium," *IEEE Transactions on Communications*, vol. 63, no. 4, pp. 1483–1495, April 2015.

[4] X. Zhang, Z. Tan, S. Xu, and C. Tao, "Utility maximization based on cross-layer design for multi-service in macro-femto heterogeneous networks," *IEEE Transactions on Wireless Communications*, vol. 12, no. 11, pp. 5607–5619, November 2013.

[5] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "TCP extensions for multipath operation with multiple addresses," RFC 6824, Internet Engineering Task Force, 2013.

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

[7] C. Raiciu, M. Handly, and D. Wischik, "Coupled congestion control for multipath transport protocols," RFC 6356, Internet Engineering Task Force, 2011.

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

[9] T. A. Le, C. seon Hong, and E.-N. Huh, "Coordinated TCP Westwood congestion control for multiple paths over wireless networks," in *Proc. of the 26th International Conference on Information Networking (ICOIN)*, 2012, pp. 92–96.

[10] Y. Cao, M. Xu, X. Fu, and E. Dong, "Explicit multipath congestion control for data center networks," in *Proc. of the 9th ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2013, pp. 73–84.

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

[12] Q. Peng, A. Walid, J. Hwang, and S. Low, "Multipath TCP: Analysis, design, and implementation," *Networking, IEEE/ACM Transactions on*, vol. PP, no. 99, pp. 1–1, 2015.

[13] C. Paasch, "Improving multipath TCP," Ph.D. dissertation, UCLouvain / ICTEAM / EPL, 2014.

[14] W. Liao, C.-J. Kao, and C.-H. Chien, "Improving tcp performance in mobile networks," *IEEE Transactions on Communications*, vol. 53, no. 4, pp. 569–571, April 2005.

[15] M. Trivellato and N. Benvenuto, "State control in networked control systems under packet drops and limited transmission bandwidth," *IEEE Transactions on Communications*, vol. 58, no. 2, pp. 611–622, February 2010.

[16] T.-A. Le, "Improving the performance of multipath congestion control over wireless networks," in *Proc. of the IEEE International Conference on Advanced Technologies for Communications (ATC)*, 2013, pp. 60–65.

[17] H.-Y. Hsieh and R. Sivakumar, "pTCP: an end-to-end transport layer protocol for striped connections," in *Proc. of the 10th IEEE International Conference on Network Protocols (ICNP)*, 2002, pp. 24–33.

[18] J. Iyengar, P. Amer, and R. Stewart, "Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths," *IEEE/ACM Transactions on Networking*, vol. 14, no. 5, pp. 951–964, 2006.

[19] C.-M. Huang and C.-H. Tsai, "WiMP-SCTP: Multi-path transmission using stream control transmission protocol (SCTP) in wireless networks," in *Proc. of the 21st IEEE International Conference on Advanced Information Networking and Applications Workshops (AINAW)*, 2007, pp. 209–214.

[20] C. Casetti and W. Gaiotto, "Westwood SCTP: load balancing over multipaths using bandwidth-aware source scheduling," in *Proc. of the 60th IEEE Vehicular Technology Conference (VTC)*, 2004, pp. 3025–3029.

[21] M. Zhang, J. Lai, A. Krishnamurthy, L. Peterson, and R. Wang, "A transport layer approach for improving end-to-end performance and robustness using redundant paths," in *Proc. of the USENIX Annual Technical Conference (ATC)*, 2004.

[22] Q. Peng, A. Walid, and S. H. Low, "Multipath TCP algorithms: Theory and design," *ACM SIGMETRICS Performance Evaluation Review*, vol. 41, no. 1, pp. 305–316, 2013.

[23] J. Oliveira, H. Salgado, and M. Rodrigues, "A new mse channel estimator optimized for nonlinearly distorted faded ofdm signals with applications to radio over fiber," *IEEE Transactions on Communications*, vol. 62, no. 8, pp. 2977–2985, Aug 2014.

[24] C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi, and R. Wang, "TCP Westwood: End-to-end congestion control for wired/wireless networks," *Wireless Networks*, vol. 8, no. 5, pp. 467–479, Sep. 2002.

[25] C. P. Fu and S. Liew, "TCP Veno: TCP enhancement for transmission over wireless access networks," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 2, pp. 216–228, 2003.

[26] S. Low, "A duality model of TCP and queue management algorithms," *IEEE/ACM Transaction Networking*, vol. 11, no. 4, pp. 525–536, 2003.

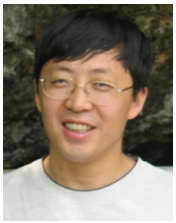
[27] K. Zhang and C. P. Fu, "Fluid-based analysis of TCP Veno with RED," in *Proc.of the IEEE Global Telecommunications Conference (GLOBECOM)*, 2006, pp. 1–5.

[28] P. Vo, A. Le, and C. Hong, "The successive approximation approach for multi-path utility maximization problem," in *Proc. of the IEEE International Conference on Communications (ICC)*, 2012, pp. 1255–1259.

[29] B. Zhou, C. P. Fu, C. Lau, and C. H. Foh, "An enhancement of TFRC over wireless networks," in *Proc. of IEEE Wireless Communications and Networking Conference (WCNC)*, 2007, pp. 3019–3024.

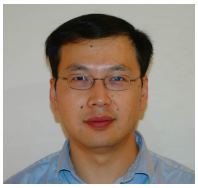
[30] Q. Pang, S. Liew, C. P. Fu, W. Wang, and V. Li, "Performance study of TCP Veno over WLAN and RED router," in *Proc.of the IEEE Global Telecommunications Conference (GLOBECOM)*, vol. 6, 2003, pp. 3237–3241.

Pingping Dong received her B.S., M.S. and Ph.D degree in Communications Engineering from Central South University, P. R. China. Currently she is a teacher in the Department of Computer Education, Hunan Normal University, Changsha, P.R. China. Her research interests are focused on protocol optimization and protocol design in wide area networks (WANs) and wireless local area networks (WLANs).



Jianxin Wang received his B.S. and M.S. degree in Computer Science from Central South University of Technology, P. R. China, and his PhD degree in Computer Science from Central South University. Currently,he is the vice dean and a professor in School of Information Science and Engineering, Central South University, Changsha, Hunan, P.R. China. He is currently serving as the executive editor of International Journal of Bioinformatics Research and Applications and serving in the editorial board of International Journal of Data Mining and Bioinformatics. He has also served as the program committee member for many international conferences. He was a program committee co-chair for the 7th, 8th, and 10th International Symposium on Bioinformatics Research and Applications (ISBRA 2011, ISBRA2012 and ISBRA2014). His current research interests include algorithm analysis and optimization, parameterized algorithm, bioinformatics and computer network. He has published more than 200 papers in various International journals and refereed conferences. He is a senior member of the IEEE.

Jiawei Huang obtained his PhD (2008) and Masters degrees (2004) from the School of Information Science and Engineering at Central South University. He also received his Bachelors (1999) degree from the School of Computer Science at Hunan University. He is now an associate professor in the School of Information Science and Engineering at Central South University, China. His research interests include performance modeling, analysis, and optimization for wireless networks and data center networks.



Haodong Wang is an Assistant Professor of Computer and Information Science at Cleveland State University. He received his Ph. D in Computer Science at College of William and Mary in Aug 2009. He also holds his Master of Science in Electrical Engineering from Penn State University, University Park, PA, and Bachelor of Engineering in Electronic Engineering from Tsinghua University, Beijing, China. Before joining Cleveland State University, He was an Assistant Professor in the Department of Math and Computer Science at Virginia State University.

Geyong Min is a Professor of High Performance Computing and Networking in the Department of Mathematics and Computer Science within the College of Engineering, Mathematics and Physical Sciences at the University of Exeter, United Kingdom. He received the PhD degree in Computing Science from the University of Glasgow, United Kingdom, in 2003, and the B.Sc. degree in Computer Science from Huazhong University of Science and Technology, China, in 1995. His research interests include Future Internet, Computer Networks, Wireless Communications, Multimedia Systems, Information Security, High Performance Computing, Ubiquitous Computing, Modelling and Performance Engineering.