Revisiting Weighted AIMD-based Congestion Control: A Comprehensive Perspective

Jiaqi Zheng, Changrong Wu, Tiancheng Lan, Chen Tian, Guihai Chen
State Key Laboratory for Novel Software Technology, Nanjing University, China

***Abstract*—Weighted congestion control aims to provide end-to-end differentiated bandwidth allocation. MulTCP and EWTCP are two closely related schemes for this purpose and they both want to achieve weighted proportionality through modifying AIMD behaviors. In this paper, we revisit the performance of MulTCP and EWTCP in terms of weighted proportionality. Through testbed experiments, we reveal a lot of counter-intuitive phenomena for achieving weighted proportionality — the switch buffer size, propagation delay and ACK options have a dominant impact on the weighted proportionality. Specifically, we develop WCC, a fundamental weighted AIMD-based congestion control building block, which can be implemented via individually modifying AI (WCC-AI) or MD (WCC-MD) behavior. We analyze WCC using extended fluid models, NS3 simulations and Linux kernel implementations with droptail and RED queues, and point out the determinant of performance. Finally, we clarify the influences of dynamic network characteristics on weighted proportionality with sufficient experimental results and summarize a basic law of how to implement weighted AIMD-based congestion control.**

I. Introduction

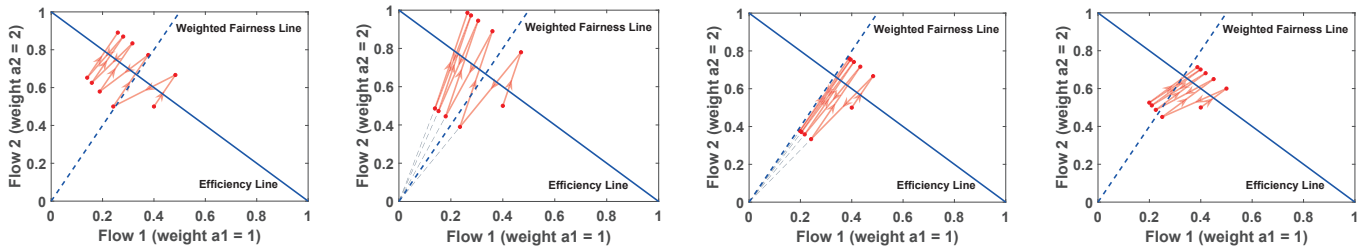
AIMD is a dominant congestion control principle in TCP, which combines the basic functionality of linear growth and exponential reduction, *i.e.*, AIMD uses additive increase to probe the usable bandwidth once receiving a ACK and multiplicative decrease to throttle the bandwidth consumption upon detecting congestion. The TCP family, such as Tahoe [19], Reno [20], Vegas [6], NewReno [16], uses AIMD principle to adjust the window size (*e.g.* sending rate), where the additive increase (multiplicative decrease) parameter of each flow is the same. Accordingly, each flow can react to potential network congestion signals — packet drops [14], [20] or RTT variants [6] or both [12], [49] — and adopt its flow to dynamic network characteristics. The AIMD's nature guarantees that each flow can be provably convergent to fairness [15], *i.e.*, each flow would obtain the equal bandwidth if they shared a single bottleneck link and had equal RTT.

However, the connection-level fairness may not always be desired in practice. First, end-to-end video transport requires QoE fairness [24], [35] as the connection-level fairness is blind to user experience. The transport protocol needs to be able to achieve weighted fairness, *i.e.*, the allocated bandwidth for each video client should be proportional to a dynamic weight computed by a algorithm that can optimize the overall utility of the video system without changes to the network. Second, under the payment model in data centers [50], the bandwidth should be proportionally divided among the tenants according to their payments [40], *i.e.*, the tenant would receive correspondingly higher bandwidth if it paid more money. Third, multipath TCP allows a single data stream to be split as multiple subflows. The allocated bandwidth of each subflow needs to be proportional to a given weight so that the entire data stream's fairness is guaranteed [46], [48].

Existing work supporting differentiated bandwidth allocation in the network level can be implemented in the switch such as WFQ [10], [39] and Diffserv [36]. Recently, NUMFabric [34] and Faircloud [40] both rely on WFQ in the programmable switches [13], [44] to enforce weighted fairness, *i.e.*, imposing different dequeue rates or packet dropping mechanisms via different queueing schedule policies. The increasingly enhanced functionality in the switch can aid to achieve weighted fairness, but this actually enforces additional constraints on the network. As the end hosts have to respond to the network feedback in time, purely end-to-end rate adjustment mechanisms still play a fundamental role in achieving weighted congestion control. From an end-to-end perspective, we ask a fundamental question whether we can modify AIMD's behavior such that a TCP flow with weight a_i can roughly receive a_i times the throughput of a TCP flow with unit weight ? MulTCP [9] and EWTCP [17] modify AIMD's behavior to achieve differentiated bandwidth allocation in the end host. Based on the TCP throughput model's analysis [38], [42], MulTCP modifies the increase and decrease parameter to a_i and $\frac{1}{2 \cdot a_i}$, respectively, and EWTCP only modifies the increase parameter to a_i^2. However, they both fail to take dynamic queuing delay into account and we will reveal that MulTCP and EWTCP can only work well in the presence of shallow-buffered switches (Sec. II), *i.e.*, the case that the window size fluctuation for different flows is desynchronized. On the contrary, the deep-buffered switches can lead to the synchronized window size fluctuation and thus the performance of MulTCP and EWTCP is far away from the perfect weighted proportionality at this point (Sec. II). This synchronized case is beyond their models' analysis. Essentially, they both ignore the significant impact from dynamic network characteristics — switch buffer size, propagation delay, the ACK options.

In this paper, we develop WCC, a fundamental weighted AIMD-based congestion control building block, which can be implemented via individually modifying AI (WCC-AI) or MD (WCC-MD) behavior. Accordingly, we derive an extended fluid model that takes the dynamic queueing delay into account. Furthermore, we use Poincaré map technique [26] to analyze the steady state behavior of WCC and conclude that

2023 IEEE/ACM 31st International Symposium on Quality of Service (IWQoS) | 979-8-3503-9973-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/IWQOS57198.2023.10188753



(a) The AIMD variations that two MulTCP flows with weight a_1 and a_2 share a single bottleneck link of capacity one.

(b) The AIMD variations that two EWTCP flows with weight a_1 and a_2 share a single bottleneck link of capacity one.

(c) The AIMD variations that two WCC flows (modify AI phase) with weight a_1 and a_2 share a single bottleneck link of capacity one.

(d) The AIMD variations of two WCC flows (modify MD phase) with weight a_1 and a_2 share a single bottleneck link of capacity one.

Fig. 1: The two-source AIMD variations comparison using the technique of Chiu and Jain [8]. Each axis corresponds to the window size of each source that is normalized to a number between 0 and 1 for convenience. If the point colored red lies in the blue solid line, this represents that one unit capacity is fully utilized, *i.e.*, the sum of the obtained bandwidth for two flows is equal to one. If the point colored red lies in the blue dashed line, this indicates that the obtained bandwidth ratio for two flows is $a_1 : a_2$. The intersection of the blue solid line and dashed line is the optimal point to achieve efficiency and weighted proportionality.

the corresponding fluid models have a fixed point or a periodic solution. Specifically, for n WCC-AI flows with weight a_1, a_2, $\cdots$, a_n, their throughput ratio is $\sqrt{a_1} : \sqrt{a_2} : \cdots : \sqrt{a_n}$ when the buffer size approaches to $\sqrt{2a_1} + \sqrt{2a_2} + \cdots + \sqrt{2a_n}$. As the buffer size becomes large enough, the throughput ratio gradually becomes $a_1 : a_2 : \cdots : a_n$. For n WCC-MD flows with weight a_1, a_2, $\cdots$, a_n, their throughput ratio is $\sqrt{a_1} : \sqrt{a_2} : \cdots : \sqrt{a_n}$ when the buffer size approaches to $\sqrt{2a_1} + \sqrt{2a_2} + \cdots + \sqrt{2a_n}$. As the buffer size becomes large enough, the throughput ratio becomes $1 : 1 : \cdots : 1$. The throughput is in general proportional to the weight for WCC-AI and WCC-MD flows.

Our concern in this paper aims to promote the understanding of weighted congestion control. We conduct extensive experiments in NS3 simulations and Linux kernel implementations to justify our theoretical analysis results. We found that the shallow/deep buffer size, more/less numbers of flows and smaller/larger propagation delay can lead to desynchronized/synchronized window size fluctuation, which is an essential reason to affect weighted proportionality. Finally, we conduct extensive experiments to quantify the weighted proportionality in terms of different buffer sizes, propagation delay, the ACK options and the total number of flows.

II. Prior Work on Weighted Congestion Control

Under weighted fairness model, when two AIMD-based TCP flows with weight a_1 and a_2 share a single bottleneck link of capacity C, the expected bandwidth for them should be $\frac{a_1}{a_1+a_2}C$ and $\frac{a_2}{a_1+a_2}C$, respectively, *i.e.*, their received bandwidth should be proportional to their weights in the steady state. Weighted congestion control is more general since the default AIMD-based TCP can be viewed as a special case that its weight equals one. In this section, we first review two representative previous works, MulTCP [9] and EWTCP [17]. We show their vulnerabilities through experimentations and point out the limitations of their theoretical basis.

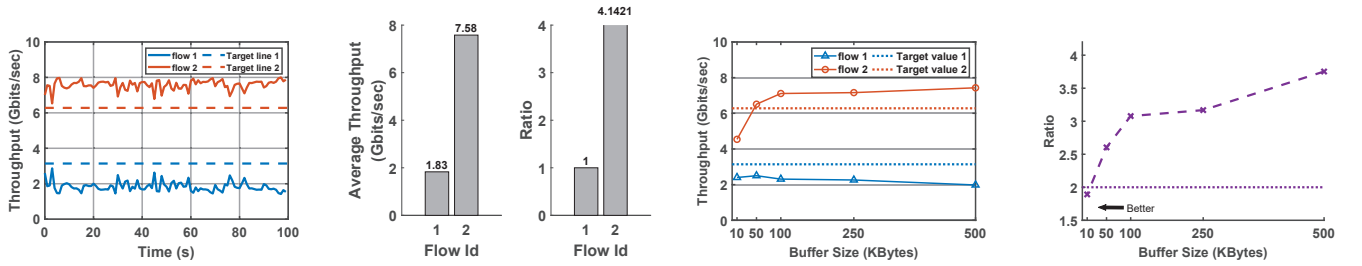
MulTCP: MulTCP modifies both AI and MD phase to achieve weighted congestion control. The increase and decrease parameter is a_i and $\frac{1}{2 \cdot a_i}$, respectively, where a_i is the corresponding weight. MulTCP-sender's behavior with weight a_i can be summarized as follows.

- **Additive Increase:** $cwnd \leftarrow cwnd + \frac{a_i}{cwnd}$
  For each ACK, the congestion window $cwnd$ increases by $\frac{a_i}{cwnd}$.
- **Multiplicative Decrease:** $cwnd \leftarrow \left(1 - \frac{1}{2 \cdot a_i}\right) \cdot cwnd$
  For each loss, the window size is decreased by $\frac{1}{2 \cdot a_i} \cdot cwnd$ when the window size $cwnd$ is larger than the slow start threshold $ssthresh$.

Now we explain the basic idea of MulTCP. Each TCP flow would obtain $\frac{1}{n}$ bandwidth when n AIMD-based TCP flows competed on a single bottleneck link. Hence, a flow that imitates behaviors of a_i flows can obtain $\frac{a_i}{n}$ bandwidth of the single bottleneck link capacity. Based on the above intuition, MulTCP is designed to make a TCP flow with weight a_i behave as if it is a collection of a_i TCP flows. However, the design doesn't align with the intuition using the technique of Chiu and Jain [8] as shown in Fig. 1(a), where the trend of the sample path from two MulTCP flows is far away from the optimal weighted proportionality. This demonstrates that MulTCP cannot work well for the highly synchronized window size fluctuations. Furthermore, MulTCP's design derives from the sawtooth model [42] that cannot take dynamic queuing delays into account. We will reveal that MulTCP can only work well in the presence of shallow-buffered switches.

EWTCP: EWTCP only modifies AI phase to achieve weighted congestion control. The increase parameter is a_i^2, where a_i is the corresponding weight. EWTCP-sender's behavior with weight a_i can be summarized as follows.

- **Additive Increase:** $cwnd \leftarrow cwnd + \frac{a_i^2}{cwnd}$
  For each ACK, the congestion window $cwnd$ increases by $\frac{a_i^2}{cwnd}$.

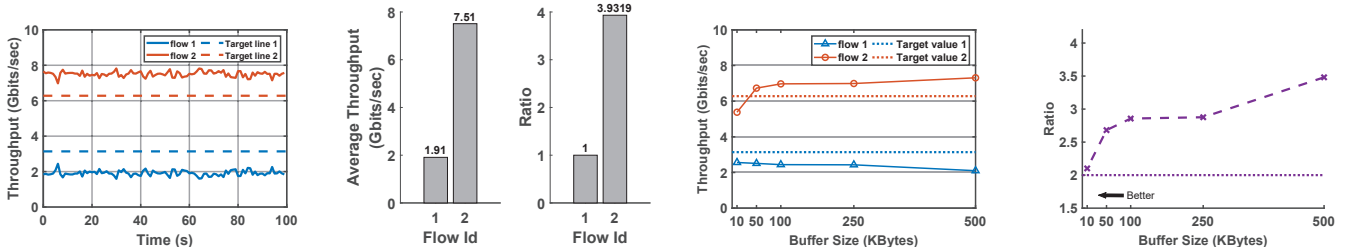


(a) The throughput trace of MulTCP with 2MBytes DropTail queues.

(b) The summary data of MulTCP with 2MBytes DropTail queues.

(c) The throughput variations of two MulTCP flows with the change of buffer size.

(d) The throughput ratio variations of two MulTCP flows with the change of buffer size.



(e) The throughput trace of EWTCP with 2MBytes DropTail queues.

(f) The summary data of EWTCP with 2MBytes DropTail queues.

(g) The throughput variations of two EWTCP flows with the change of buffer size.

(h) The throughput ratio variations of two EWTCP flows with the change of buffer size.

Fig. 2: Weighted Proportionality of two MulTCP and EWTCP flows with weight 1 and 2 in our testbed, which consists of one Mellanox SN2700 switch associated with two hosts as the sources and one host as the destination. The link capacity is 10 Gbps.

- **Multiplicative Decrease:** $cwnd \leftarrow \frac{cwnd}{2}$
  For each loss, the window size is cut down by half.

EWTCP is also called WAIMD [17], where a_i^2 in the AI phase is derived from a classic theoretical model that can be used to estimate TCP throughput [38]. However, this model cannot take dynamic queuing delays into account and uses a static estimated average RTT instead. We will reveal that EWTCP can only work well in the presence of shallow-buffered switches.

Discussions: Fig. 2 illustrates our testbed experiments under different buffer sizes. Fig. 2(a) and Fig. 2(e) show the weighted proportionality in the presence of the deep-buffered switches, where the buffer size is 2 MBytes. We can observe that the average throughput for two MulTCP flows with weight 1 and 2 is 1.83 Gbps and 7.58 Gbps, respectively, where the throughput ratio is around 1:4 and is far away from the optimal point 1:2 as shown in Fig. 2(b). Similarly, the average throughput for two EWTCP flows with weight 1 and 2 is 1.91 Gbps and 7.51 Gbps as shown in Fig. 2(f), which also has an obvious deviation from the optimal weighted proportionality.

What's more, after we take numerous tests for MulTCP and EWTCP, we find that the switch buffer size greatly have an impact on the throughput ratio of MulTCP and EWTCP flows. For example, Fig. 2(c) and Fig. 2(g) reveal the throughput variations of MulTCP and EWTCP flows with different configurations of switch buffer size. Specifically, the throughput for two MulTCP flows is 2.46 Gbps and 4.58 Gbps when the buffer size equals 10 KB, while the throughput becomes 2.38 Gbps and 7.13 Gbps when the buffer size equals 100 KB. The throughput for two EWTCP flows is 2.55 Gbps and 5.38 Gbps when the buffer size equals 10 KB, while the throughput becomes 2.42 Gbps and 6.96 Gbps when the buffer size equals 100 KB. Fig. 2(d) and Fig. 2(h) show the normalized results, where MulTCP and EWTCP approach the target ratio only when the buffer size is less than 10 KB. This demonstrates that the weighted proportionality of MulTCP and EWTCP is perfect in the presence of shallow-buffered switches. In general, MulTCP and EWTCP are far from the optimal weighted proportionality. We will show and discuss more about this in the following sections.

III. Weighted AIMD-based Congestion Control

In this section, we present WCC, a weighted AIMD-based congestion control building block, which can be implemented via individually modifying AI or MD phase.

A. WCC Intuition and Design

Our intuition comes from the technique of Chiu and Jain [8], where two WCC flows with weight 1 and 2 compete with each other and adjust their window sizes according to the augmented AIMD principle, leading to a sample path shown in Fig. 1(c) and Fig. 1(d). Specifically, Fig. 1(c) shows the throughput fluctuation for two WCC flows when we modify AI phase, while Fig. 1(d) shows that when we modify MD phase. We can observe that they both fluctuate around the optimal weighted proportionality point (*i.e.*, the intersection between the blue solid line and the blue dashed line) once get converged. Hence, we summary this intuition as the following two algorithms: WCC-AI and WCC-MD.

WCC-AI: WCC-AI only modifies AI phase to achieve weighted congestion control. The increase parameter is a_i,

where a_i is the corresponding weight. WCC-AI's behavior with weight a_i can be summarized as follows.

- **Additive Increase:** $cwnd \to cwnd + \frac{a_i}{cwnd}$
  For each ACK, the congestion window $cwnd$ increases by $\frac{a_i}{cwnd}$.
- **Multiplicative Decrease:** $cwnd \to \frac{cwnd}{2}$
  For each loss, the window size is cut down by half.

WCC-MD: WCC-MD only modifies MD phase to achieve weighted congestion control. The decrease parameter is $\frac{1}{2\cdot a_i}$, where a_i is the corresponding weight. WCC-MD's behavior with weight a_i can be summarized as follows.

- **Additive Increase:** $cwnd \to cwnd + \frac{1}{cwnd}$
  For each ACK, the congestion window $cwnd$ increases by $\frac{1}{cwnd}$.
- **Multiplicative Decrease:** $cwnd \to (1 - \frac{1}{2\cdot a_i}) \cdot cwnd$
  For each loss, the window size is decreased by $\frac{1}{2\cdot a_i} \cdot cwnd$.

B. An Extended Fluid Model

Without loss of generality, we consider two long-lived WCC flows that traverses a single bottleneck link with capacity C. The weight of these two WCC-sender flows is a_1 and a_2, respectively. The extended fluid models for WCC-AI and WCC-MD under droptail queues are omitted due the space limitation.

WCC with droptail queues: The fluid model (1) and (2) capture the dynamics of window size $W_1(t)$, $W_2(t)$ and the queue length $q(t)$ for WCC-AI and WCC-MD flows, respectively. We use the indicator variable $p(t) = \mathbb{1}_{\{q(t)>K\}}$ to characterize the packet dropping behavior at the switch, *i.e.*, the packets will be dropped once the queue length is beyond K. The RTT can be captured by the equation $R(t) = d + \frac{q(t)}{C}$, where d is a fixed propagation delay.

$$\begin{cases} \frac{dW_1}{dt} = \frac{a_1}{R(t)} - \frac{W_1(t)W_1(t-R^*)}{2R(t-R^*)}p(t-R^*), \\ \frac{dW_2}{dt} = \frac{a_2}{R(t)} - \frac{W_2(t)W_2(t-R^*)}{2R(t-R^*)}p(t-R^*), \\ \frac{dq}{dt} = \frac{W_1(t)+W_2(t)}{R(t)} - C. \end{cases} \tag{1}$$

The first two equations in fluid model (1) capture the window size variations with time t for two WCC-AI flows. They consist of an additive term $\frac{a_i}{R(t)}$ that is proportional to the flow weight a_i and a multiplicative decrease term $\frac{W_i(t)W_i(t-R^*)}{2R(t-R^*)}$ when $p(t-R^*)$ equals one, where $R^* = d + \frac{K}{C}$. The third equation in fluid model (1) characterizes the queue length evolution, which is the difference between the arrival rate $\frac{W_1(t)+W_2(t)}{R(t)}$ at t and the departure rate C.

$$\begin{cases} \frac{dW_1}{dt} = \frac{1}{R(t)} - \frac{1}{2a_1}\cdot\frac{W_1(t)W_1(t-R^*)}{R(t-R^*)}p(t-R^*), \\ \frac{dW_2}{dt} = \frac{1}{R(t)} - \frac{1}{2a_2}\cdot\frac{W_2(t)W_2(t-R^*)}{R(t-R^*)}p(t-R^*), \\ \frac{dq}{dt} = \frac{W_1(t)+W_2(t)}{R(t)} - C. \end{cases} \tag{2}$$

For the fluid model (2), the condition $a_i > \frac{1}{2}$ $(i = 1, 2)$ must hold since the decrease parameter is at most one. The first two equations capture the window size variations with time t for two WCC-MD flows. And the third equation characterizes the queue length evolution with time t. The additive increase term is $\frac{1}{R(t)}$, which is a default TCP setting like NewReno [16]. The multiplicative decrease term is proportional to $\frac{1}{2a_1}$.

We rewrite the fluid model (1) and (2) by replacing the variables $R^* = d + \frac{K}{C}$, $\widetilde{W}(t) = W(R^*t)$ and $\widetilde{q}(t) = \frac{q(R^*t)-K}{CR^*}$. Accordingly, $p(t)$ can be rewritten as $\widetilde{p}(t)$, *i.e.*, $\widetilde{p}(t) = \mathbb{1}_{\{\widetilde{q}(t)>0\}}$. $\bar{w}$ is defined as $\bar{w} = Cd + K$. Since the queue does not underflow, we define the variable $\psi = \frac{K}{Cd}$ and the fluid models can be rewritten as (3) and (4). Here the system has five parameters (C, d, K, a_1, a_2) and the dynamics can be determined by $\bar{w}$, a_1 and a_2.

$$\begin{cases} \frac{d\widetilde{W_1}}{dt} = \frac{a_1}{1+\widetilde{q}(t)} - \frac{\widetilde{W_1}(t)\widetilde{W_1}(t-1)}{2(1+\widetilde{q}(t-1))}\widetilde{p}(t-1), \\ \frac{d\widetilde{W_2}}{dt} = \frac{a_2}{1+\widetilde{q}(t)} - \frac{\widetilde{W_2}(t)\widetilde{W_2}(t-1)}{2(1+\widetilde{q}(t-1))}\widetilde{p}(t-1), \\ \frac{d\widetilde{q}}{dt} = \begin{cases} \frac{1}{\bar{w}}\frac{\widetilde{W_1}(t)+\widetilde{W_2}(t)}{1+\widetilde{q}(t)} - 1 & \widetilde{q}(t) > \frac{-\psi}{1+\psi}, \\ \max\left(\frac{1}{\bar{w}}\frac{\widetilde{W_1}(t)+\widetilde{W_2}(t)}{1+\widetilde{q}(t)} - 1, 0\right) & \widetilde{q}(t) = \frac{-\psi}{1+\psi}. \end{cases} \end{cases} \tag{3}$$

$$\begin{cases} \frac{d\widetilde{W_1}}{dt} = \frac{1}{1+\widetilde{q}(t)} - \frac{1}{2a_1}\cdot\frac{\widetilde{W_1}(t)\widetilde{W_1}(t-1)}{1+\widetilde{q}(t-1)}\widetilde{p}(t-1), \\ \frac{d\widetilde{W_2}}{dt} = \frac{1}{1+\widetilde{q}(t)} - \frac{1}{2a_2}\cdot\frac{\widetilde{W_2}(t)\widetilde{W_2}(t-1)}{1+\widetilde{q}(t-1)}\widetilde{p}(t-1), \\ \frac{d\widetilde{q}}{dt} = \begin{cases} \frac{1}{\bar{w}}\frac{\widetilde{W_1}(t)+\widetilde{W_2}(t)}{1+\widetilde{q}(t)} - 1 & \widetilde{q}(t) > \frac{-\psi}{1+\psi}, \\ \max\left(\frac{1}{\bar{w}}\frac{\widetilde{W_1}(t)+\widetilde{W_2}(t)}{1+\widetilde{q}(t)} - 1, 0\right) & \widetilde{q}(t) = \frac{-\psi}{1+\psi}. \end{cases} \end{cases} \tag{4}$$

Now we begin to discuss the existence of the possible fixed point in the fluid model (3) and (4). For fluid model (3), if this fixed point exists, $(\widetilde{W_1}, \widetilde{W_2}, \widetilde{q})$ must satisfy the following equations.

$$\frac{a_1}{1+\widetilde{q}(t)} - \frac{\widetilde{W_1}(t)\widetilde{W_1}(t-1)}{2(1+\widetilde{q}(t-1))}\widetilde{p}(t-1) = 0 \tag{5}$$

$$\frac{a_2}{1+\widetilde{q}(t)} - \frac{\widetilde{W_2}(t)\widetilde{W_2}(t-1)}{2(1+\widetilde{q}(t-1))}\widetilde{p}(t-1) = 0 \tag{6}$$

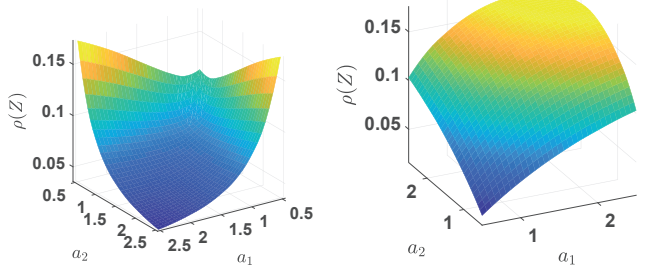
$$\frac{1}{\bar{w}}\frac{\widetilde{W_1}(t)+\widetilde{W_2}(t)}{1+\widetilde{q}(t)} - 1 = 0 \tag{7}$$

The equations (5), (6) and (7) have solution if and only if the condition $\bar{w} \le \sqrt{2a_1} + \sqrt{2a_2}$ holds. We define $\omega' = \sqrt{2a_1} + \sqrt{2a_2}$ and discuss the solution in cases:

(i) $\bar{w} \le \omega'$: the fluid model (3) has a unique fixed point $(\widetilde{W_1}, \widetilde{W_2}, \widetilde{q}) = (\sqrt{2a_1}, \sqrt{2a_2}, \frac{\omega'}{\bar{w}} - 1)$, which means that the fluid model (1) has a fixed point $(\sqrt{2a_1}, \sqrt{2a_2}, \omega' - Cd)$. The equation $\bar{w} \le \omega'$ implies that $K \le \sqrt{2a_1} + \sqrt{2a_2} - Cd$, which indicates to the case when the value of K is smaller, *i.e.*, the switch buffer size is shallow.

(ii) $\bar{w} > \omega'$: the fluid model has a periodic solution or limit cycle, instead of a fixed point. This implies that $K > \sqrt{2a_1} + \sqrt{2a_2} - Cd$ and corresponds to the case when the value of K becomes larger, *i.e.*, the switch buffer size is deep.

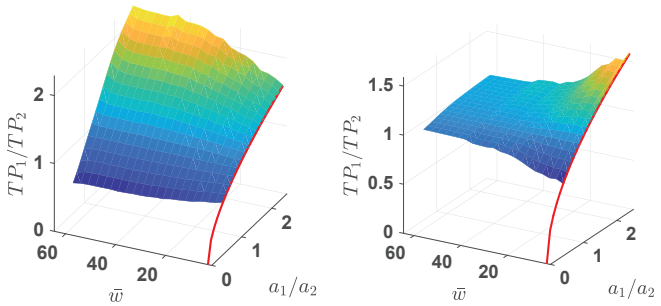
Similar to the discussion in fluid model (3), the fluid model (4) has similar conclusions.



(a) The spectral radius (the z-axis) variations with weight a_1 and a_2 when $\bar{w}$ equals to 5.

(b) The spectral radius (the z-axis) variations with weight a_1 and a_2 when $\bar{w}$ equals to 5.

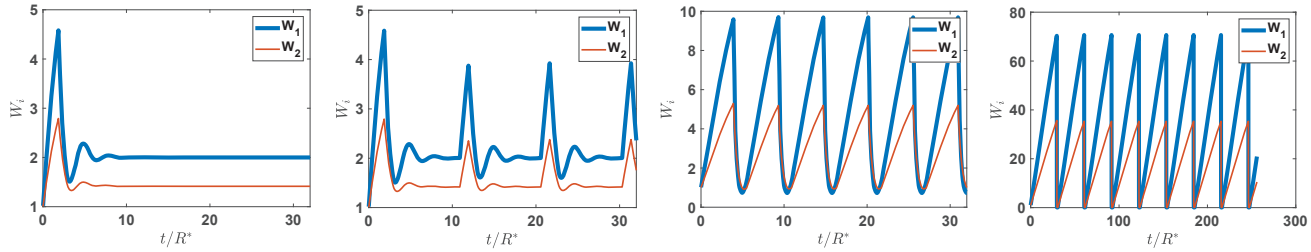
Fig. 3: The spectral radius is always less than one, *i.e.*, $\rho(Z) < 1$.



(a) The throughput ratio (the z-axis) variations for two WCC flows in fluid model (3).

(b) The throughput ratio (the z-axis) variations for two WCC flows in fluid model (4).

Fig. 4: The throughput ratio for two WCC flows in the fluid model.



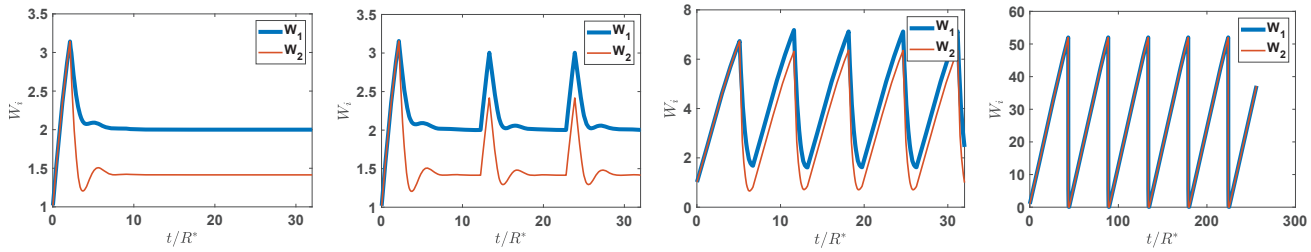
(a) The window size variations when $\bar{w}$ equals to $2+\sqrt{2}$.

(b) The window size variations when $\bar{w}$ equals to $2+\sqrt{2}+0.01$.

(c) The window size variations when $\bar{w}$ equals to 10.

(d) The window size variations when $\bar{w}$ equals to 100.

Fig. 5: The window size variations for two WCC flows in fluid model (3), where a_1 equals to two and a_2 equals to one.



(a) The window size variations when $\bar{w}$ equals to $2/\sqrt{3}+\sqrt{2}$.

(b) The window size variations when $\bar{w}$ equals to $2/\sqrt{3}+\sqrt{2}+0.01$.

(c) The window size variations when $\bar{w}$ equals to 10.

(d) The window size variations when $\bar{w}$ equals to 100.

Fig. 6: The window size variations for two WCC flows in fluid model (4), where a_1 equals to two and a_2 equals to one.

C. Steady State Analysis

We denote $x(t) = (\widetilde{W}_1(t), \widetilde{W}_2(t), \widetilde{q}(t))^T$ as the phase diagram in the fluid model (3) and (4). Accordingly,

$$\dot{x}(t) = F(x(t), u(t-1)), u(t) = \mathbb{1}_{\{cx(t)>0\}} \tag{8}$$

where $c = [0, 0, 1]$ and $u(t) = \widetilde{p}(t)$.

We use Poincaré map to study the stability of limit cycles. We define the switching plane as $S = \{x \in \mathbb{R}^3 : cx = 0\}$ and let $S^+ = \{x \in \mathbb{R}^3 : cx > 0\}$ and $S^- = \{x \in \mathbb{R}^3 : cx < 0\}$. The limit cycle passes through the switching plane S twice in each period, once from S^+ and once from S^-.

To affiliate the proof of the theorem, we first give the following notations. Let $x^*(t)$ denote the trajectory of the limit cycle of the equations (8). Assume that $x^*(t)$ traverses the switching plane from S^+ to S^- at time $t_0 = 0$, *i.e.*, $x^*(0) = x^*$, and the period of the limit cycle is T. Let

$$Z = \left(I - \frac{F(x^*, 1)c}{cF(x^*, 1)}\right) \exp\left(\int_0^T J_F(x^*(s), u(s-1))ds\right)$$

where x^* traverses the switching plane from S^+ to S^-at time $t = 0$. where I is the identity matrix and J_F is the Jacobian matrix of F with respect to x. The integral of the matrix J_F is term-by-term integration and the exponential function $\exp(\cdot)$ is the matrix exponential. Suppose $cF(x^*, 1) \neq 0$, i.e., $x^*(t)$ is nontangent with the switching plane S at the traversing points. Note that Z is 3×3 matrices.

Lemma 1. *There exists $\epsilon > 0$ such that any trajectory starting from $x = x^* + \Delta x \in B_\epsilon(x^*) \cap S$ will intersect and traverse*

S, and the second traversing point $x(T+\Delta t)$ satisfies

$$x\left(T+\Delta t\right)-x^{*}=Z\Delta x+O\left(\Delta^{2}\right)$$

Proof. There exists some $\epsilon>0$ such that the trajectory starting from $x=x^{*}+\Delta x\in B_{\epsilon}(x^{*})\cap S$ will traverse S a second time at $T+\Delta t$. Define $\Delta x(t)=x(t)-x^{*}(t)$. Then,

$$\begin{aligned}\dot{\Delta x}(t)&=F(x(t),u(t-1))-F\left(x^{*}(t),u(t-1)\right)\\&=F\left(x^{*}(t)+\Delta x(t),u(t-1)\right)-F\left(x^{*}(t),u(t-1)\right)\\&=J_{F}\left(x^{*}(t),u(t-1)\right)\Delta x(t)+O\left(\Delta^{2}\right)\end{aligned}$$

Since $\Delta x(0)=\Delta x$, we get

$$\begin{aligned}&x(T+\Delta T)-x^{*}(T+\Delta T)\\&=\exp\left(\int_{0}^{T}J_{F}\left(x^{*}(s),u(s-1)\right)ds\right)\Delta x+O\left(\Delta^{2}\right)\end{aligned}$$

Making a series expansion in ΔT, we get:

$$x^{*}(T+\Delta T)-x^{*}=F(x^{*},1)\Delta T+O\left(\Delta^{2}\right)$$

Since both $x(T+\Delta T)$ and x^{*} are on the switching plane S, we can derive that $cx(T+\Delta T)=cx^{*}=0$. Accordingly,

$$\Delta T=-\frac{c\exp\left(\int_{0}^{T}J_{F}\left(x^{*}(s),u(s-1)\right)ds\right)\Delta x}{cF\left(x^{*},1\right)}+O\left(\Delta^{2}\right)$$

Combining the following equation, we conclude the proof.

$$\begin{aligned}&x\left(T+\Delta T\right)-x^{*}=\exp\left(\int_{0}^{T}J_{F}\left(x^{*}(s),u(s-1)\right)ds\right)\Delta x\\&+F\left(x^{*},1\right)\Delta T+O\left(\Delta^{2}\right)\end{aligned}$$

□

Theorem 1. *The Poincaré map and its associated limit cycle is locally asymptotically stable if and only if*

$$\rho(Z)<1,$$

where $\rho(\cdot)$ is the spectral radius.

Proof. Note that the conditions $x(T+\Delta t)=P(x)$ and $x^{*}=P(x^{*})$ can be established, where $P(\cdot)$ is the Poincaré map. From Lemma 1, we have

$$P(x)=P(x^{*})+Z\Delta x+O\left(\Delta^{2}\right)$$

Consequently, the Jacobian of the Poincaré map at x^{*} is Z. Therefore, the limit cycle is locally asymptotically stable if and only if $\rho(Z)<1$. □

Since we cannot obtain the analytical solution of the function $F(\cdot)$, the matrix Z is unknown. Hence, we have to use numerical solution to determine the spectral radius. The experiment results show that $\rho(Z)$ is always less than one, indicating WCC can be provably convergent to a steady state. Due to space limits, we only show parts of the results in Fig. 3. The window size variations with the parameter $\bar{w}$ for the WCC-AI and WCC-MD flows are shown in Fig. 5 and Fig. 6. Accordingly, we can derive the following observations from Fig. 4.

We first define the throughput TP_i of a flow i as follows.

$$TP_{i}=\frac{1}{T}\int_{0}^{T}\frac{1}{\bar{w}}\frac{\widetilde{W_{i}}(t)}{1+\widetilde{q}(t)}dt,\qquad i=1,2,\cdots,n$$

Observation 1. *The throughput ratio of two WCC-AI flows in fluid model* (3) *can be derived as follows.*

$$\frac{TP_{1}}{TP_{2}}\rightarrow\begin{cases}\frac{\sqrt{a_{1}}}{\sqrt{a_{2}}} & \bar{w}\rightarrow\sqrt{2a_{1}}+\sqrt{2a_{2}},\\ \frac{a_{1}}{a_{2}} & \bar{w}\rightarrow\infty.\end{cases}\tag{9}$$

where a_1 and a_2 are the weights of two flows, respectively.

Observation 2. *The throughput ratio of two WCC-MD flows in fluid model* (4) *can be derived as follows.*

$$\frac{TP_{1}}{TP_{2}}\rightarrow\begin{cases}\frac{\sqrt{a_{1}}}{\sqrt{a_{2}}} & \bar{w}\rightarrow\sqrt{2a_{1}}+\sqrt{2a_{2}},\\ 1 & \bar{w}\rightarrow\infty.\end{cases}\tag{10}$$

where a_1 and a_2 are the weights of two flows, respectively.

Furthermore, we can extend the above observations to a more general case for n flows.

Observation 3. *When $\bar{w}\rightarrow\sqrt{2a_{1}}+\sqrt{2a_{2}}+\cdots+\sqrt{2a_{n}}$, the throughput ratio of n WCC-AI flows can be $TP_{1}:TP_{2}:\cdots:TP_{n}\rightarrow\sqrt{a_{1}}:\sqrt{a_{2}}:\cdots:\sqrt{a_{n}}$. When $\bar{w}\rightarrow\infty$, the throughput ratio of n WCC-AI flows can be $TP_{1}:TP_{2}:\cdots:TP_{n}\rightarrow\sqrt{a_{1}}:a_{1}:a_{2}:\cdots:a_{n}$. where a_1, a_2, $\cdots$, a_n are the weights of n flows, respectively.*

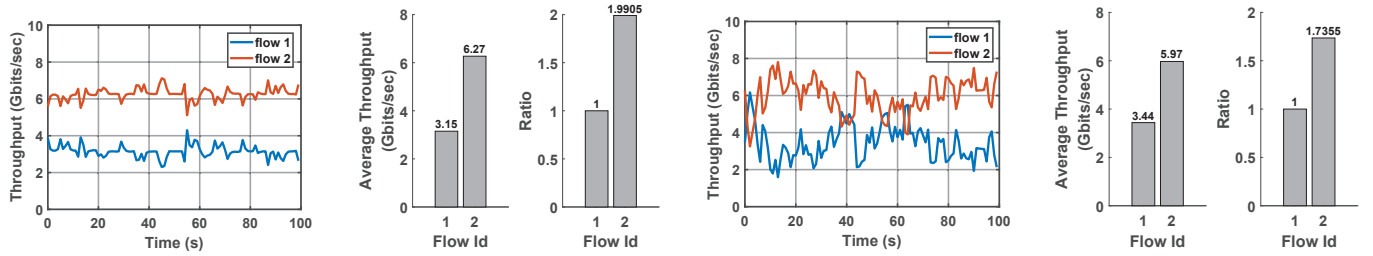
Observation 4. *When $\bar{w}\rightarrow\sqrt{2a_{1}}+\sqrt{2a_{2}}+\cdots+\sqrt{2a_{n}}$, the throughput ratio of n WCC-MD flows can be $TP_{1}:TP_{2}:\cdots:TP_{n}\rightarrow\sqrt{a_{1}}:\sqrt{a_{2}}:\cdots:\sqrt{a_{n}}$. When $\bar{w}\rightarrow\infty$, the throughput ratio of n WCC-MD flows can be $TP_{1}:TP_{2}:\cdots:TP_{n}\rightarrow 1$. where a_1, a_2, $\cdots$, a_n are the weights of n flows, respectively.*

Based on the observations and analysis above, we conclude that modifying AI phase can well achieve weighted proportionality. At the same time, different buffer sizes and the total number of flows can quantitatively affect the weighted proportionality.

IV. Implementation and Testbed Experiments

We have implemented WCC-AI, MulTCP and EWTCP as Linux congestion control modules. We use them to conduct testbed experiments and find that the results are consistent with our theoretical analysis. In this section, we present our experimental results and summarize the basic principles of achieving weighted congestion control.

Testbed Setup: We implement WCC-AI, MulTCP and EWTCP all based on the standard Linux kernel congestion control API. Our testbed consists of one Mellanox SN2700 switch associated with two hosts as the sources and one host as the destination, *i.e.*, a typical star topology. Two TCP flows with weight 1 and 2 are generated by Iperf3 [18] and share a single bottleneck link of 10 Gbps capacity. The testbed is built inside a rack where servers are connected by optical fibers with a fixed length of around 5 meters, so the propagation delay of the links can be roughly calculated by dividing fiber



(a) The throughput trace of WCC-AI with DropTail queues and selective ACK.

(b) The summary data of WCC-AI with DropTail queues and selective ACK.

(c) The throughput trace of WCC-AI with RED queues and selective ACK.

(d) The summary data of WCC-AI with RED queues and selective ACK.

(e) The throughput trace of WCC-AI with DropTail queues and cumulative ACK.

(f) The summary data of WCC-AI with DropTail queues and cumulative ACK.

(g) The throughput trace of WCC-AI with RED queues and cumulative RED.

(h) The summary data of WCC-AI with RED queues and cumulative RED.

Fig. 7: The weighted proportionality of two WCC-AI flows with weight 1 and 2 under droptail and RED queues in our testbed. The buffer size of droptail queues are 2 MBytes. The minimum threshold and maximum threshold of RED queues are 1 MBytes and 5MBytes respectively.

length with light speed, *i.e.*, 17ns. The measured throughput for each TCP flow is reported by Iperf3. We use Drop-tail and RED queues in Mellanox SN2700 switch to observe WCC-AI's behaviors with different queue management mechanisms. In terms of the TCP ACK mechanisms, we exploit the Linux kernel parameter `net.ipv4.tcp_sack` to switch between the selective ACK [30] and the cumulative ACK and observe their influence on the weighted proportionality.

Experiment Results: WCC-AI can achieve near perfect weighted proportionality. The measured throughput variations for each flow are shown in Fig. 7(a). We can observe that WCC-AI can provide differentiated bandwidth allocation, while maintaining the stability in a long term. Furthermore, the throughput ratio is approximately equal to the ratio of their weights, *i.e.*, $1:2$. Likewise, from Fig. 7(c), we can see that under the RED condition, the throughput of two WCC-AI flows maintain a ratio $1:2$ roughly most of the time, although the random drop causes more oscillations. However, we also notice that only when using the selective ACK mechanism, can WCC-AI achieve such good performance. We will take a more specific inspections about this in the Sec. V.

As we stated in Sec. III, the decisive factor of weighted proportionality is the parameter $\bar{w}$. The $\bar{w}$ is the average congestion window in the steady state, which is dominated by the bandwidth delay product. That is $\bar{w} \propto C \cdot d + K$, where K is the buffer size, C is the link capacity and d is the propagation delay. Hence, we change the buffer size K and the link propagation delay d respectively to observe their influence on the weighted proportionality.

TABLE I: The throughput ratio for two WCC-AI flows with weight 1 and 2 varies with the switch buffer size.

Buffer size (Bytes)	Average throughput (Gbps)		Ratio
	Flow 1	Flow 2	
12000 K	3.12	6.31	1 : 2.022
5000 K	3.11	6.31	1 : 2.029
2000 K	3.15	6.27	1 : 1.990
1000 K	3.22	6.20	1 : 1.925
500 K	3.49	5.93	1 : 1.699
250 K	3.69	5.61	1 : 1.520
100 K	3.22	5.12	1 : 1.590

TABLE II: The throughput ratio for two WCC-AI flows with weight 1 and 2 varies with the propagation delay.

Emulated delay	Average throughput (Gbps)		Ratio
	Flow 1	Flow 2	
5 ms	1.20	2.37	1 : 1.975
1 ms	2.17	4.15	1 : 1.912
500 μs	2.69	5.15	1 : 1.914
375 μs	2.88	5.20	1 : 1.806
250 μs	2.86	5.61	1 : 1.962
100 μs	3.27	5.68	1 : 1.737
50 μs	3.42	5.80	1 : 1.696
10 μs	3.57	5.75	1 : 1.611
1 μs	3.61	5.77	1 : 1.598

The switch buffer size has a significant impact on the weighted proportionality. We perform a set of experiments to measure the weighted proportionality variations with the switch buffer size K. We can observe that the experiment results in Tab. I are consistent with our theoretical analysis in Sec. III. When the buffer size is small, *i.e.*, $\bar{w}$ approaches $\sqrt{2a_1} + \sqrt{2a_2} \simeq 3.414$, the ratio keeps fluctuating around the expected value $1 : \sqrt{2} \simeq 1 : 1.414$. With the growth of

the buffer size, the throughput ratio of two WCC-AI flows increases to the perfect value $1:2$. In general, the weighted proportionality of AIMD-based weighted congestion control mechanisms have two relatively stable bounds: when the switch buffer size increases, the ratio of two WCC-AI flows with weight 1 and 2 will move towards an upper bound $1:2$. Otherwise, it will move towards a lower bound $1:\sqrt{2}$. The results in Fig.2 also demonstrates the same fact.

The propagation delay also has a significant impact on the weighted proportionality. According to our theoretical analysis, the throughput ratio is not only related to the switch buffer size which dominates the queuing delay, but also related to the propagation delay. Thus, we run another set of experiments in our testbed to observe the performance of WCC-AI with the propagation delay variations. Since the propagation delay of physical links are fixed and hard to change, we use the `netem` module in Linux Traffic Control to emulate the propagation delay between senders and receivers. In this set of experiments, the switch buffer size is fixed at 500 KBytes. From Tab. II, we can see that when the propagation delay increases, the throughput ratio of two WCC-AI flows approach the perfect value $1:2$ gradually. That demonstrates that the throughput ratio moves from the lower bound to the upper bound, when the incremental delay makes $\bar{w}$ become larger.

V. Simulation And Analysis

Setup: We exploit NS-3 simulator [37] and implement the weighted congestion controls — MulTCP, EWTCP and WCC-AI — as subclasses of `TcpNewReno`. The link capacity is set to 10 Gbps, the link delay is set to 20 μs and the switch buffer size is set to 200 KBytes. The selective ACK is enabled.

Experiment Results: WCC-AI can achieve weighted proportionality under dynamic network characteristics. As shown in Fig. 8, we firstly investigate the weighted proportionality of WCC-AI compared with MulTCP and EWTCP in a dynamic traffic environment. The four flows have the weight 4, 3, 2 and 1, respectively. We can see that EWTCP has very poor stability and is far away from perfect weighted proportionality. MulTCP and WCC-AI can well converge to a steady state in a long run while WCC-AI performs much better than MulTCP. In Fig. 8(c), we can observe that the throughput ratio for WCC-AI flows approaches the perfect ratio even in a network with dynamic traffic. When the first flow and the third flow coexist in the time intervals between 2s and 4s, the throughput ratio is precisely close to $3:1$. While all four flows together are in the network during the time intervals from 4s to 7s, WCC-AI converges quickly and approaches the perfect ratio within half a second. Similarly, WCC-AI also performs well when only the second flow and the fourth flow are in the network.

The selective ACK and cumulative ACK can also significantly affect the weighted proportionality. We investigate the performance of NewReno, MulTCP, EWTCP and WCC-AI with two different ACK mechanisms. The comparison is illustrated in Fig. 9 and Tab. III shows the average throughput ratio with 5 s duration corresponding to Fig. 9. We can observe that the ACK options do not dramatically affect the fairness of NewReno in the long run: the average throughput ratio of five flows is 1:1.01:1.01:1.03:1.02 with the selective ACK and 1:1.04:1.07:0.74:0.92 with the cumulative ACK. However, the weighted proportionality of MulTCP, EWTCP and WCC-AI perform better when the selective ACK is enable. More specifically, we find that using cumulative ACK is easily to make CWND fluctuate sharply and potentially limits the congestion window size in the long term. As a result, the weighted proportionality performs poorly with the cumulative ACK. Our testbed experiments in Fig. 7 verify this fact as well.

TABLE III: The average throughput ratio with 5 s duration corresponding to Fig.9.

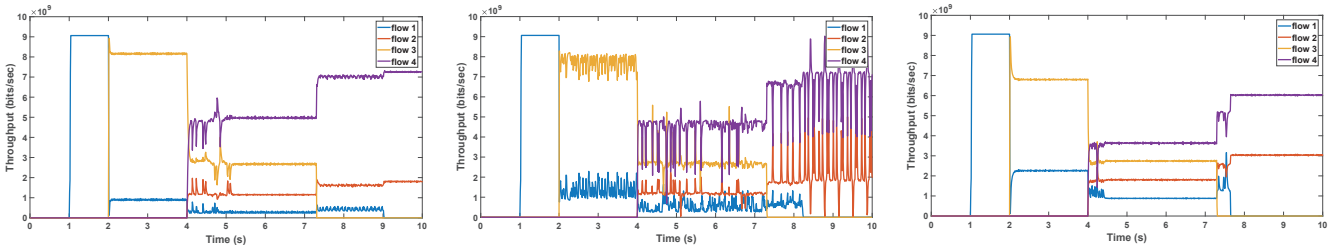
CC Scheme	Average throughput ratio of five flows	
	SACK	Non-SACK
NewReno [16]	1:1.01:1.01:1.03:1.02	1:1.04:1.07:0.74:0.92
MulTCP [9]	1:3.67:8.81:16.03:24.96	1:1.10:1.36:1.54:1.55
EWTCP [17]	1:2.38:5.30:9.37:13.21	1:2.81:3.15:3.28:3.53
WCC-AI	1:1.65:2.50:3.33:4.14	1:1.44:2.48:2.56:3.10

VI. Related Work

Control Principles: Under a synchronized-feedback assumption, Chiu and Jain [8] analyze classic linear controls — AIMD, AIAD, MIAD, MIMD — in terms of efficiency and fairness and conclude that only AIMD can achieve efficiency and fairness simultaneously in the steady state. Subsequently, RAP [41], RLM [31], LDA [43], Reno [20], Vegas [6], NewReno [16] and so on are proposed to use AIMD as its control logic. Later, GAIMD [47], binomial congestion control [4] and SIMD [23] are proposed to generalize the linear controls with non-linear controls.

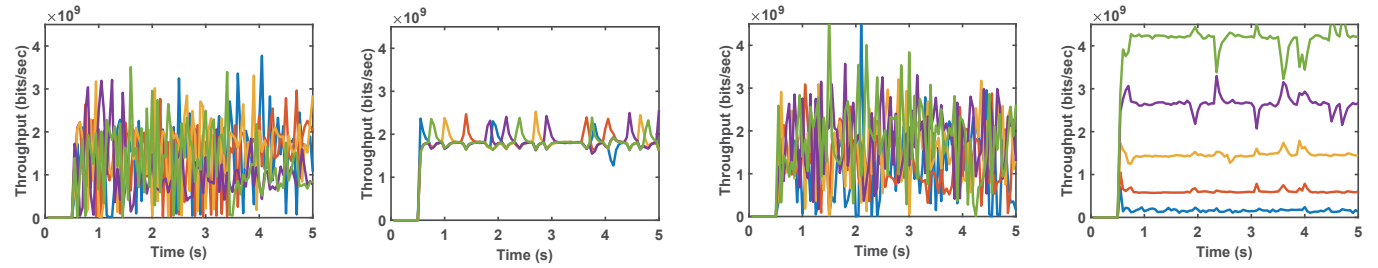
Congestion Control: TCP leverages network feedback such as packet drops [14], [16], [20] or RTT variants [3], [6], [7] or both [49] in response to potential network congestion. DCTCP [1], [2] combines the ECN marking mechanism to perform rate reduction in MD phase, providing smoother transmission rate in data centers. For RDMA-enabled network, DCQCN [51], Timely [32] and HPCC [27] rely on the feedback of programmable switches to facilitate the congestion control. Remy [45] and PCC Vivace [11] leverage machine learning technique to perform fine-grained congestion control.

Weighted Fairness: As the network evolves from a free-use model to a pay-for-use model, the fairness principle [5], [21], [25], [33] can be increasingly augmented with weighted fairness, *i.e.*, differentiated bandwidth allocation. The network-level weighted fairness relies on WFQ [10], [39] and Diffserv [36] techniques implemented in the switch. NUMFabric [34], DGD [28] and Faircloud [40] rely on WFQ to enforce weighted fairness. In a purely end-to-end perspective, MulTCP [9] and EWTCP [17] modify AIMD's behavior to perform weighted congestion control. A recent work [35] implements weighted congestion control based on CUBIC [14] and FAST [22] and aims to achieve weighted QoE fairness. In addition, multipath TCP [29], [46] requires weighted conges-

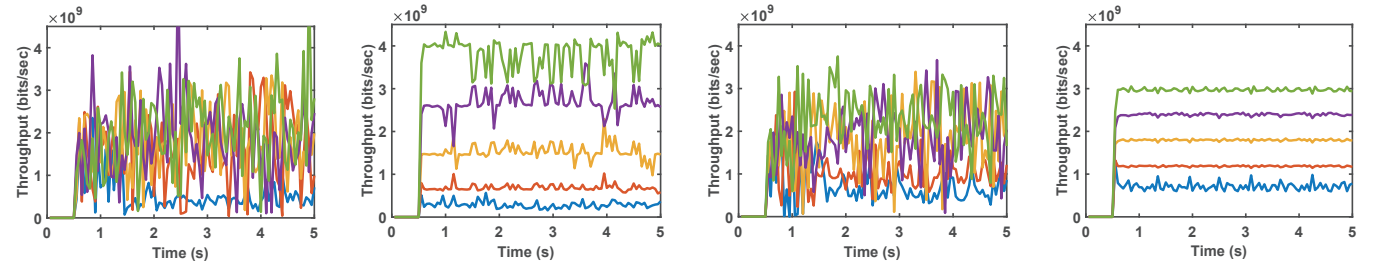


(a) MulTCP (b) EWTCP (c) WCC-AI

Fig. 8: The weighted proportionality comparison for MulTCP, EWTCP, and WCC-AI with the selective ACK. The purple, yellow, red, blue line corresponds to the flow with weight 4, 3, 2 and 1, respectively.



(a) The performance of NewReno with cumulative ACK (b) The performance of NewReno with selective ACK. (c) The performance of MulTCP with cumulative ACK (d) The performance of MulTCP with selective ACK.



(e) The performance of EWTCP with cumulative ACK (f) The performance of EWTCP with selective ACK. (g) The performance of WCC-AI with cumulative ACK (h) The performance of WCC-AI with selective ACK.

Fig. 9: The influence of different ACK options, *i.e.* selective ACK or accumulative ACK. The green, purple, yellow, red, blue line corresponds to the flow with weight 5, 4, 3, 2 and 1, respectively.

tion control [17] to adjust the rate of each subflow such that the entire data stream's fairness is guaranteed.

VII. Conclusion

In this paper, we pinpoint that existing works fail to reveal the hidden relation between the weighted proportional performance and different network parameters. We design a fundamental weighted congestion control mechanism WCC and use the fluid model to analyse the critical factors for achieving weighted proportionality. Testbed experiments and NS3 simulation results can well match our analysis. We believe that our work can give a much deeper understanding in terms of weighted congestion control and affiliate developing novel weight congestion control mechanisms in the future.

VIII. Acknowledgement

We would like to thank anonymous reviewers for their valuable comments. This work was supported in part by National Key R&D Program of China (2022YFB2901800), China NSF grants (62172206) and China NSF of Jiangsu Province (BK20201248).

References

[1] Mohammad Alizadeh, Albert G. Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. Data center tcp (dctcp). In *Proc. ACM SIGCOMM*, volume 40, pages 63–74, 2010.

[2] Mohammad Alizadeh, Adel Javanmard, and Balaji Prabhakar. Analysis of dctcp: stability, convergence, and fairness. In *Proc. ACM SIGMETRICS*, volume 39, pages 73–84, 2011.

[3] Venkat Arun and Hari Balakrishnan. Copa: Practical delay-based congestion control for the internet. In *Proc. USENIX NSDI*, pages 329–342, 2018.

[4] D. Bansal and H. Balakrishnan. Binomial congestion control algorithms. In *Proc. IEEE INFOCOM*, volume 2, pages 631–640, 2001.

[5] Dimitri P Bertsekas, Robert G Gallager, and Pierre Humblet. *Data networks*, volume 2. Prentice-Hall International New Jersey, 1992.

[6] Lawrence S. Brakmo and Larry L. Peterson. Tcp vegas: end to end congestion avoidance on a global internet. *IEEE Journal on Selected Areas in Communications*, 13(8):1465–1480, 1995.

[7] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. Bbr: congestion-based congestion control. *Proc. ACM Queue*, 60(2):58–66, 2016.

[8] Dahming Chiu and Raj Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and Isdn Systems*, 17(1):1–14, 1989.

[9] Jon Crowcroft and Philippe Oechslin. Differentiated end-to-end internet services using a weighted proportional fair sharing tcp. *ACM SIGCOMM Computer Communication Review*, 28(3):53–69, 1998.

[10] Alan J. Demers, Srinivasan Keshav, and Scott Shenker. Analysis and simulation of a fair queueing algorithm. In *Proc. ACM SIGCOMM*, volume 19, pages 1–12, 1989.

[11] Mo Dong, Tong Meng, Doron Zarchy, Engin Arslan, Yossi Gilad, Brighten Godfrey, and Michael Schapira. Pcc vivace: Online-learning congestion control. In *Proc. USENIX NSDI*, pages 343–356, 2018.

[12] Zhuoxuan Du, Jiaqi Zheng, Hebin Yu, Lingtao Kong, and Guihai Chen. A unified congestion control framework for diverse application preferences and network conditions. In *Proc. ACM CoNEXT*, 2021.

[13] Yong Feng, Zhikang Chen, Haoyu Song, Wenquan Xu, Jiahao Li, Zijian Zhang, Tong Yun, Ying Wan, and Bin Liu. Enabling in-situ programmability in network data plane: From architecture to language. In *Proc. USENIX NSDI*, pages 635–649, 2022.

[14] Sangtae Ha, Injong Rhee, and Lisong Xu. Cubic: a new tcp-friendly high-speed tcp variant. *Operating Systems Review*, 42(5):64–74, 2008.

[15] Go Hasegawa, Masayuki Murata, and Hideo Miyahara. Fairness and stability of congestion control mechanisms of tcp. In *Proc. IEEE INFOCOM*, volume 3, pages 1329–1336. IEEE, 1999.

[16] T. Henderson, S. Floyd, A. Gurtov, and Y. Nishida. The newreno modification to tcp's fast recovery algorithm. *RFC 6582*, 2012.

[17] Michio Honda, Yoshifumi Nishida, Lars Eggert, Pasi Sarolahti, and Hideyuki Tokuda. Multipath congestion control for shared bottleneck. In *Proc. PFLDNeT Workshop*, 2009.

[18] iPerf. *https://iperf.fr/*.

[19] Van Jacobson. Congestion avoidance and control. *ACM SIGCOMM computer communication review*, 18(4):314–329, 1988.

[20] Van Jacobson. Modified tcp congestion avoidance algorithm. *Email to the end2end-interest mailing list*, 1990.

[21] Jeffrey Jaffe. Bottleneck flow control. *IEEE Transactions on Communications*, 29(7):954–962, 1981.

[22] Cheng Jin, David X. Wei, and Steven H. Low. Fast tcp: motivation, architecture, algorithms, performance. In *Proc. IEEE INFOCOM*, volume 4, pages 2490–2501. IEEE, 2004.

[23] Shudong Jin, Liang Guo, I. Matta, and A. Bestavros. Tcp-friendly simd congestion control and its convergence behavior. In *Proc. IEEE ICNP*, pages 156–164, 2001.

[24] Ri Lu Kai Xiao Songlin Li Jufeng Chen Jingyu Yang Chunli Zong Aiyun Chen Qinghua Wu Chen Sun Gareth Tyson Jinyang Li, Zhenyu Li and Hongqiang Harry Liu. Livenet: A low-latency video transport network for large-scale live streaming. In *Proc. ACM SIGCOMM*, pages 1–14, 2022.

[25] Frank Kelly. Charging and rate control for elastic traffic. *European transactions on Telecommunications*, 8(1):33–37, 1997.

[26] HASSAN K. KHALIL. *Nonliear Systems*. Prentice Hall, 2002.

[27] Yuliang Li, Rui Miao, Hongqiang Harry Liu, Yan Zhuang, Fei Feng, Lingbo Tang, Zheng Cao, Ming Zhang, Frank Kelly, Mohammad Alizadeh, and Minlan Yu. Hpcc: high precision congestion control. In *Proc. ACM SIGCOMM*, pages 44–58, 2019.

[28] S. H. Low and D. E. Lapsley. Optimization flow control. i. basic algorithm and convergence. *IEEE/ACM Transactions on Networking*, 7(6):861–874, 1999.

[29] Yuanwei Lu, Guo Chen, Bojie Li, Kun Tan, Yongqiang Xiong, Peng Cheng, Jiansong Zhang, Enhong Chen, and Thomas Moscibroda. Multi-path transport for rdma in datacenters. In *Proc. USENIX NSDI*, pages 357–371, 2018.

[30] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. Tcp selective acknowledgment options. *RFC 2018*, 1996.

[31] Steven McCanne, Van Jacobson, and Martin Vetterli. Receiver-driven layered multicast. pages 117–130, 1996.

[32] Radhika Mittal, Nandita Dukkipati, Emily Blem, Hassan Wassel, Monia Ghobadi, Amin Vahdat, Yaogong Wang, David Wetherall, and David Zats. Timely: Rtt-based congestion control for the datacenter. In *Proc. ACM SIGCOMM*, pages 537–550, 2015.

[33] Jeonghoon Mo and Jean Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on networking*, (5):556–567, 2000.

[34] Kanthi Nagaraj, Dinesh Bharadia, Hongzi Mao, Sandeep Chinchali, Mohammad Alizadeh, and Sachin Katti. Numfabric: Fast and flexible bandwidth allocation in datacenters. In *Proc. ACM SIGCOMM*, pages 188–201, 2016.

[35] Vikram Nathan, Vibhaalakshmi Sivaraman, Ravichandra Addanki, Mehrdad Khani, Prateesh Goyal, and Mohammad Alizadeh. End-to-end transport for video qoe fairness. In *Proc. ACM SIGCOMM*, pages 408–423, 2019.

[36] Kathleen M. Nichols, Steven Blake, Fred Baker, and David L. Black. Definition of the differentiated services field (DS field) in the ipv4 and ipv6 headers. *RFC 2474*, 1998.

[37] ns3 Network Simulator. *https://www.nsnam.org/*.

[38] Jitendra Padhye, Victor Firoiu, Donald F. Towsley, and James F. Kurose. Modeling tcp throughput: a simple model and its empirical validation. In *Proc. ACM SIGCOMM*, pages 303–314, 1998.

[39] Abhay K. Parekh and Robert G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, 1993.

[40] Lucian Popa, Gautam Kumar, Mosharaf Chowdhury, Arvind Krishnamurthy, Sylvia Ratnasamy, and Ion Stoica. Faircloud: sharing the network in cloud computing. In *Proc. ACM SIGCOMM*, pages 187–198, 2012.

[41] R. Rejaie, M. Handley, and D. Estrin. Rap: An end-to-end rate-based congestion control mechanism for realtime streams in the internet. In *Proc. IEEE INFOCOM*, volume 3, pages 1337–1345, 1999.

[42] Floyd Sally and Kevin Fall. Router mechanisms to support end-to-end congestion control. *Technical report*, 1997.

[43] Dorgham Sisalem and Henning Schulzrinne. The loss-delay based adjustment algorithm: A tcp-friendly adaptation scheme. In *Proc. NOSSDAV*, 1998.

[44] Anirudh Sivaraman, Suvinay Subramanian, Mohammad Alizadeh, Sharad Chole, Shang-Tse Chuang, Anurag Agrawal, Hari Balakrishnan, Tom Edsall, Sachin Katti, and Nick McKeown. Programmable packet scheduling at line rate. In *Proc. ACM SIGCOMM*, pages 44–57, 2016.

[45] Keith Winstein and Hari Balakrishnan. Tcp ex machina: computer-generated congestion control. In *Proc. ACM SIGCOMM*, volume 43, pages 123–134, 2013.

[46] Damon Wischik, Costin Raiciu, Adam Greenhalgh, and Mark Handley. Design, implementation and evaluation of congestion control for multipath tcp. In *Proc. USENIX NSDI*, volume 11, pages 8–8, 2011.

[47] Y.R. Yang and S.S. Lam. General aimd congestion control. In *Proc. IEEE ICNP*, pages 187–198, 2000.

[48] Hebin Yu, Jiaqi Zheng, Zhuoxuan Du, and Guihai Chen. Mplibra: Complementing the benefits of classic and learning-based multipath congestion control. In *Proc. IEEE ICNP*, 2021.

[49] Gaoxiong Zeng, Wei Bai, Ge Chen, Kai Chen, Dongsu Han, Yibo Zhu, and Lei Cui. Congestion control for cross-datacenter networks. In *Proc. IEEE ICNP*, pages 1–10. IEEE, 2019.

[50] Jiaqi Zheng, Hong Xu, Guihai Chen, and Haipeng Dai. Minimizing transient congestion during network update in data centers. In *Proc. IEEE ICNP*, 2015.

[51] Yibo Zhu, Haggai Eran, Daniel Firestone, Daniel Firestone, Chuanxiong Guo, Marina Lipshteyn, Yehonatan Liron, Jitendra Padhye, Shachar Raindel, Mohamad Haj Yahia, and Ming Zhang. Congestion control for large-scale rdma deployments. In *Proc. ACM SIGCOMM*, pages 523–536, 2015.