RateMP: Optimizing Bandwidth Utilization with High Burst Tolerance in Data Center Networks

Jiangping Han*, Kaiping Xue*†, Wentao Wang‡, Ruidong Li§, Qibin Sun*, Jun Lu*‡
*School of Cyber Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China
‡Department of EEIS, University of Science and Technology of China, Hefei, Anhui 230027, China
§National Institute of Information and Communications Technology, Kanazawa University, Tokyo 184-0015, Japan
†Corresponding author: K. Xue (kpxue@ustc.edu.cn)

***Abstract*—Load balancing in data center networks (DCNs) is a crucial and complex undertaking. Multi-path TCP (MPTCP) has been proposed as a cost-effective solution that aims to distribute workloads and improve network resource utilization. However, it can escalate buffer occupancy and undermine burst tolerance, particularly in scenarios involving incast short flows. To address these limitations, we propose a novel multi-path congestion control algorithm, RateMP, to optimize bandwidth utilization efficiency while ensuring burst tolerance in DCNs. RateMP employs a hybrid window and rate control loop with coupled gradient projection adjustment, enabling fast and fine-grained bandwidth allocation and accelerating convergence. Additionally, RateMP eliminates the limitation of cwnd with under-rate pacing to protect incast and busty flows. We prove that RateMP is Lyapunov stable and asymptotically stable, and show the improvement of RateMP through a kernel-based implementation and extended large-scale simulations. RateMP keeps high bandwidth utilization, cuts RTT by 2x and reduces flow completion times (FCT) by 45% in incast scenarios compared to existing algorithms.**

I. Introduction

Data center networks (DCNs) are the essential infrastructure [1], [2], which host a wide variety of cloud computing and storage services. With the increase in demand of distributed computing and storage resources, the data transmission overheads among the distributed servers/hosts have become the bottleneck of DCNs' performance. Therefore, improving network transmission performance is becoming a crucial issue for the development of large-scale DCNs [3], [4].

To bridge the bottleneck of data transmission, a lot of DCN topologies make dense interconnect fabrics to increase the network resource among the distributed servers/hosts [5], such as Fat-Tree, BCube, etc. In such topologies, networks expand the available links and aggregation bandwidth between every two layers (such as, between the edge layer and aggregation layer, between the edge layer and core layer) to hold more transmission capacity. Such dense interconnect fabrics increase the available network bandwidth on the whole. Then, end-to-end transmissions can disperse their data in the network and accelerate the data transmission. In this context, multi-path transmission protocols like Multi-path TCP (MPTCP) [6] offer a solution by enabling parallel transmissions over multiple paths and distributing traffic across the network [7]–[9]. By fully utilizing the idle links within DCNs, MPTCP can improve the overall bandwidth utilization efficiency, achieving up to a two-fold improvement [10]–[12].

Except the bandwidth utilization efficiency, another crucial performance metric in DCNs is achieving low flow completion time (FCT) and high burst tolerance for short-burst flows. Short-burst flows are the common communication pattern in DCNs, usually occurring as incast traffic where multiple nodes concurrently send data to a single destination for distributed high-performance applications [13]. However, MPTCP demonstrates undesirable performance for short-burst flows, creating a dilemma. That is to say, utilizing multiple paths in each connection spreads data transmission over as many as possible available links in the network to fully utilize the idle network resource, but at the same time, inevitably causes a rise in the resource occupation while meeting burst and incast flows at local bottlenecks and leads to low burst tolerance. Existing coupled congestion control algorithms designed for MPTCP usually focus on fully utilizing the network bandwidth resource, but struggle to solve the increasingly serious problem of heavy burst and incast traffic in DCNs. On the one hand, MPTCP increases the number of active subflows in the network. When local incast and burst traffic occurs, the number of active subflows aggregated at the bottleneck is much larger than using single-path protocols (such as TCP or RDMA), leading to a rise in resource usage. On the other hand, the jitter of transmission control rises sharply when too many subflows pass through one bottleneck, again leading to incast performance degradation.

We find that the key insight for achieving high transmission performance in DCNs is to achieve optimal utilization of network bandwidth resources through multi-path transmission while avoiding the increase in delay and FCT under burst and incast situations. In this paper, we propose a novel solution of coupled congestion control algorithm to achieve the above goals, called RateMP, to optimize bandwidth utilization with high burst tolerance. RateMP improves network bandwidth utilization from two perspectives. First, RateMP utilizes a model-based gradient projection adjustment method with the consideration of network utility and subflow congestion balance to control the congestion window (cwnd) and sending rate, instead of the Increase and Multiplicative Decrease (AIMD) and Additive Increase and Additive Decrease (AIAD) method. RateMP limits the queues in the network and adjusts

the offset of the control loop according to how much the transmission rate deviates from the feedback congestion level. Through accurately perceiving the congestion degree information in the network, RateMP couples and adjusts subflows' rates to achieve the optimal network utility and subflow-level congestion balancing. Therefore, the transmission rate adjusts rapidly when the deviation from the target congestion level is large, while the transmission rate is smooth when it is close to the target congestion level. Second, RateMP utilizes a hybrid cwnd/rate control state transition, which uses pure rate settings to reduce the short-time-scale incast burstiness by removing the limitation of minimum cwnd on subflows. RateMP also utilizes a cooperative method to detect the congestion in networks, which can yield a quick convergence performance, improving the load balancing ability among MPTCP subflows and reducing the network queue delay. The contributions of the paper are summarized as follows.

- We analyze the key issues related to multi-path transmission with a dynamic multi-path resource optimization model, and show the shortcomings of the existing multi-path protocols while taking into account the unique characteristics of DCNs.
- We propose RateMP to optimize both bandwidth utilization and burst tolerance under dynamic traffic loads. RateMP introduces a hybrid multi-path control loop to incorporate the demands for maintaining short queues and expediting the dynamic convergence rate. Then, RateMP eliminates the limitation of cwnd with a dynamic state transition to protect incast and busty flows.
- We prove that RateMP is Lyapunov stable and asymptotically stable in the network. Through Linux kernel-based evaluations and extended large-scale simulation experiments, we show that RateMP outperforms the state-of-the-art congestion control algorithms, achieving low RTTs and reducing FCTs in different scenarios while keeping high bandwidth utilization.

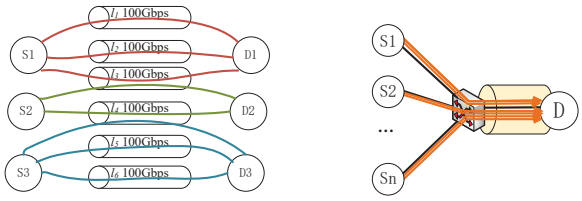
The rest of this paper is organized as follows. Section II presents the problem formulation. Sections III and IV present the design and analysis of RateMP. Section V presents the simulation results. Section VI presents a brief review of related works. Finally, Section VII concludes the paper.

II. Background and Problem Analysis

In this section, we first discuss the dilemma in which using end-to-end multi-path transmission leads to higher throughput for long flows whereas it also leads to more buffer pressure under local congestion. Then, we formulate and analyze the multi-path rate allocation problem in large-scale DCNs, and show the design goals of the proposed RateMP algorithm in this paper.

A. Dilemma of Utilizing MPTCP in DCNs

In the context of the dense paths in DCNs, utilizing multi-path can improve the overall bandwidth utilization and the network utility. We show it from a simple example in Fig. 1(a). There are six bottlenecks in the network, where each bottleneck has a bandwidth of 100 Gbps. For single-path transmission, one connection can only utilize the bandwidth on one bottleneck at once. A multi-path transmission can fully utilize the bandwidth in the network, which improves resource utilization.



(a) Non-incast (b) Incast

Fig. 1. Examples of multi-path transmissions in DCNs.

On the contrary, using multi-path transmission will lead to low burst tolerance. As shown in Fig. 1(b), a larger number of aggregated subflows adds a larger amount of data at the incast bottleneck, and the possibility for burst packet loss greatly increases. This is because each subflow keeps a minimum cwnd (usually set to the size of 1 packet). This is used to keep a subflow always holding the ability to send data. If network conditions change and paths become non-congested, a subflow can increase its rate to fully utilize the bandwidth resource of the path. However, when a large number of subflows are aggregated at one bottleneck, the queue occupancy increases a lot, leading to low performance of incast traffic. Moreover, while using multi-path transmission fully utilizes network links, it also leaves less room for burst traffic, resulting in low burst tolerance.

Therefore, existing MPTCP is difficult to achieve both high bandwidth utilization and high burst tolerance in DCNs. In the following subsections, we formulate the multi-path transmission problem and analyze the problem.

B. Problem Formulation

We define the multi-path transmission in a DCN as follows. The network is a graph that consists of a link set $\mathbb{L} = \{l_i\}$ and a flow set $\mathbb{S} = \{S_k\}$. Layered on top of the link set is a path set $\mathbb{R} = \{R_j\}$. Link l_i denotes a point-to-point link. Path R_j denotes an end-to-end routing path, which consists of one or more links. Flow S_k denotes an MPTCP flow, which consists of one or more paths. Each link l_i has a finite capacity c_i. The relationship between $\mathbb{L}$ and $\mathbb{R}$ is denoted by a matrix $\mathbf{A}$, where $a_{ij} = 1$ if $l_i \in R_j$, and $a_{ij} = 0$ otherwise. The relationship between $\mathbb{R}$ and $\mathbb{S}$ is denoted as a matrix $\mathbf{B}$, where $b_{jk} = 1$ if $R_j \in S_k$, and $b_{jk} = 0$ otherwise. The rate matrix is denoted as $\mathbf{x} = \{x_{jk}\}$, where x_{jk} is the rate of path R_j of flow S_k. Let $U(y_k)$ denote the utility function of a flow, which is strictly concave and twice continuously differentiable, and $U(0) = -\infty$. Let y_k denote the overall rate of a flow:

$$y_k = \sum_{j \in S_k} x_{jk}. \tag{1}$$

The objective is to determine the rate matrix $\mathbf{x}$ to maximize the total network utility subject to link capacity constraints:

$$\begin{aligned} \max_{\mathbf{x}\geq 0} \quad & \sum_{k} U(y_k) \\ s.t. \quad & \mathbf{ABx} \leq \mathbf{C}, \end{aligned} \tag{2}$$

where the constraint condition means that the total traffic carried on each link cannot exceed the link capacity.

Additionally, the static problem formulation can be seen as a part of the dynamic network environment. The dynamic network environment is divided into several intervals, each of which is considered a static network. The model assumes a discrete notion of time slot, $\{H\}$, denoting the times of events (flows join and leave). Given this, we can now model a dynamic network, $\mathcal{N}^H = \{\mathbb{L}^H, \mathbb{S}^H\}$, where $\mathcal{N}^H$ is the model of the network during time slot H, $\mathbb{L}^H$ and $\mathbb{S}^H$ are the link set and flow set at time slot H, respectively. Typically, the link set stays the same at each time slot, in the absence of link failures or router failures. The flow set changes at different time slots, as new flows join the network or old flows complete.

The goal of congestion control is that the network can converge quickly from any initial point to an optimal equilibrium point in any time slot. When the time slot moves from the old H to the new $H+1$, the old equilibrium point is destroyed as the network changes and becomes the initial point of the new time slot. Then, the network is readjusted and converged to the new equilibrium point. We ignore the statement of time slots in the preceding and following descriptions.

C. Problem Analysis

Let $\lambda = \{\lambda_1, \lambda_2, ...\}$ denote the Lagrangian multipliers, the dual problem of the original problem is:

$$\begin{aligned} \max_{\lambda\geq 0} D(\lambda) &= \sum_{k} D_k(\lambda) - \sum_{i} \lambda_i c_i, \\ D_k(\lambda) &= \max_{\mathbf{x}\geq 0} \left\{ U_k\left(y_k\right) - \sum_{j\in S_k} \left(\sum_{i} \lambda_i a_{ij} \right) x_{jk} \right\}. \end{aligned} \tag{3}$$

Let $p_j = \sum_{i} \lambda_i a_{ij}$, assume that at the network equilibrium point, the optimal $\mathbf{p} = \{p_j | p_j = \sum_{i\in R_j} \lambda_i\}$ is $\mathbf{p}^* = \{p_j^* | p_j^* = \sum_{i\in R_j} \lambda_i^*\}$. According to Karush-Kuhn-Tucker Conditions [14], the optimal solution of a flow k is:

$$\begin{aligned} & U'\left(y_k^*\right) = \min_{j\in S_k} p_j^*, \\ & x_{jk}^* \geq 0, \quad \text{if} \quad p_j^* = \min_{j\in S_k} p_j^*, \\ & x_{jk}^* = 0, \quad \text{if} \quad p_j^* > \min_{j\in S_k} p_j^*. \end{aligned} \tag{4}$$

A flow should transmit data only on the paths with the smallest p_j^*. For other paths whose p_j^* is larger, the flow does not transmit data on them. Window-based congestion control algorithms at least keep a minimum cwnd on subflows, which still take a high bandwidth occupancy when the subflow's RTT is small. However, this violates the rate allocation principle of multi-path transmission and causes difficulties in improving the overall utility.

When several flows pass through the same paths, they should have the same sending rate to optimize network performance. Assume that there are flows $k_1, ..., k_n$ in the network that experience the same bottleneck links $l_{i_1}, ..., l_{i_m}$. At the equilibrium of the network, the congestion degrees of bottleneck links are $\lambda_{i_1}^*, ..., \lambda_{i_m}^*$, respectively. The congestion degrees of the paths in S_{k_i} with bottlenecks $i_1, ..., i_m$ are $p_{j_1}^* = \lambda_{i_1}^*, ..., p_{j_m}^* = \lambda_{i_m}^*$, respectively. Then, we have $U'(y_{k_i}) = \min_{j\in S_{k_i}} p_j^* = \min \lambda_{i_m}^*$. Since $U(\cdot)$ is strictly concave, we have $y_{k_1} = \cdots = y_{k_i} = \cdots = y_{k_n}$. Therefore, the flows passing through the same bottlenecks should have the same rate. Moreover, in an incast scenario, all flows should achieve the same rate since they have the same $\min\{p_j^*\}$, no matter how many subflows they have.

III. RateMP Design

In this section, we provide RateMP, to achieve high global network resource utilization and high burst tolerance under dynamic traffic loads in DCNs.

A. Overview

RateMP utilizes the well-established feedback mechanism for assessing network congestion levels and employs multi-signal detection techniques to expedite the dynamic convergence rate of multi-path congestion control. This approach improves the load balancing convergence speed in dynamic network traffic, ensuring efficient utilization of available network resources. When the sending rate is small, RateMP utilizes a rate-limited control logic and adds delay detection for accelerating traffic convergence. We will primarily focus on introducing RateMP from the following aspects:

- **Fine-grained congestion detection:** RateMP utilizes a combination of two kinds of congestion detection, ECN and delay detection, to ensure load balancing and high throughput. Among them, probabilistic ECN feedback provides the main congestion detection with accurate in-network congestion feedback, delay is added to speed up the rate increase when the sending rate is low.
- **Stable and fast-convergent control loop:** RateMP utilizes a gradient projection method. It limits the queues in the network and adjusts the offset of the control loop according to the degree to which the send rate deviates from the feedback congestion level, providing less fluctuation in sending rate as well as optimizing network utility.
- **Rate control under window limitation:** RateMP sets a *pacing_rate* to directly control the pure sending rate on subflows under low rates. By removing the limitation of cwnd, RateMP can reduce the influences caused by different RTTs and make a fine-grained rate allocation.

B. Fine-grained Congestion Detection

RateMP utilizes a probabilistic ECN marking to notify the in-network congestion degree, and utilizes a delay-detection to enhance the in-network congestion degree detection under some specific situations. Firstly, the marking probability is

proportional to the queue length of the switch port queue, the in-network congestion degree is feedback by one-bit ECN with a probabilistic marking rate. Then, RateMP uses double signal detection of ECN and delay to adapt to the rapid congestion change in the network.

Probabilistic ECN-marking and congestion degree detection: RateMP sets the ECN mark associated with the queue length. By calculating the proportion of ACKs with the ECN mark of a subflow, senders can perceive the congestion degree of each subflow from the one-bit ECN. RateMP's ECN marking mechanism uses the standard ECN marking, which is supported by most switches [15]. Two thresholds K_{min} and K_{max} of queue length are set at switches. When the queue length exceeds K_{min}, packets are marked with a linearly increasing probability with a max probability P_{max}, until the queue length exceeds K_{max}. After the queue length exceeds K_{max}, all packets are marked. With a queue length $queue$, the ECN mark probability is:

$$p_{mark} = \begin{cases} 0, & queue < K_{min} \\ \frac{queue \cdot P_{max}}{K_{max} - K_{min}}, & K_{min} \leq queue < K_{max} \\ 1, & queue \geq K_{max} \end{cases} \quad (5)$$

By setting $K_{min} = 0$ and $P_{max} = 1$, the ECN mark probability is $\frac{queue}{K_{max}}$, which is proportional to the queue length. By choosing a suitable K_{max}, RateMP is able to keep the queue length within a certain range.

RateMP maintains an estimation of ECN marking probability of subflow j, denoted as β_j, which is calculated by the fraction of ECN-marked ACKs. β_j updates once per RTT:

$$\beta_j = (1 - \gamma)\beta_j + \gamma F, \quad (6)$$

where F is the ECN-marked proportion of ACKs in the recent cwnd, γ is the update parameter. We set $\gamma = 1/16$, which is similar to the setting in [16]. Let p_j denote the congestion degree of path, $p_j = \alpha\beta_j$, α is a positive constant, since p_j is in direct proportion to β_j.

Delay-based congestion degree detection: Let new_rtt_j denote the RTT of the most recent packet, $base\_rtt_j$ denote the minimum RTT of subflow j. Then, an RTT offset $\Delta rtt_j = new\_rtt_j - base\_rtt_j$ shows the queue delay, where $\Delta rtt_j = q_j/c$, and c is the link bandwidth. Then the queue length q_j on the path can be estimated as:

$$q_j = \Delta rtt_j \cdot c = (new\_rtt_j - base\_rtt_j) \cdot c. \quad (7)$$

The same as ECN-detection, the congestion degree can be estimated as $p_j = \alpha q_j / K_{max}$.

ECN detection provides an accurate notification of the in-network congestion [17]. However, the sender needs to count the percentage of ECN-marked ACKs to understand the degree of congestion on the path, where sufficient samples are needed to calculate the actual proportions. Therefore, when the sending rate is small, it takes a long time to calculate. If the link goes idle, ECN detection reacts slower. Conversely, delay detection provides a fast response to changing network congestion. A large descending RTT of a single packet can help senders quickly detect congestion changes and improve the sending rate. Therefore, RateMP utilizes ECN detection as the main method to detect congestion. When the sending rate is small, delay detection is added to provide fast rate adjustment.

C. Stable and Fast-convergent Control Loop

RateMP leverages a model-based gradient projection adjustment method to control the cwnd, taking into account network utility and subflow congestion balancing, instead of the traditional Additive Increase Multiplicative Decrease (AIMD) or Additive Increase Additive Decrease (AIAD) approaches. This method limits the queues in the network and adjusts the offset of the control loop based on the degree of deviation from the feedback congestion level. When there is a significant deviation between the estimated congestion level at the current rate and the detected congestion level, the transmission rate is rapidly adjusted to improve the convergence speed of load balancing under dynamic network traffic. Otherwise, as it approaches the detected congestion level, the transmission rate remains stable.

We show the control loop of a single flow and omit the subscript k for sake of simplicity. Let $x_j(t)$ denote the rate of subflow j, $p_j(t)$ denote the feedback congestion degree of subflow j at time t. A flow adjusts the sending rate of each subflow according to the connection rate and subflow congestion degree,

$$x_j(t+1) = \left[x_j(t) + \theta_t \left(U^{'}(y(t)) - p_j(t)\right)\right]^+, \quad (8)$$

where θ_t is the step size at time t, $y_k(t) = \sum_{j \in S_k} x_{jk}(t)$, $[x]^+ = \max\{0, x\}$.

We utilizes $U(y) = \log(y)$ to achieve proportional fairness [18], $\theta_t = \sum_j x_j(t)\alpha_{total}$, and $p_j(t) = \beta_j(t)/\alpha_{total}$. The updated rule of sending rate is:

$$x_j(t+1) = \left[x_j(t) + \sum_j x_j(t)\left(\frac{\alpha_{total}}{\sum_j x_j(t)} - \beta_j(t)\right)\right]^+. \quad (9)$$

Then, the cwnd is: $w_j(t+1) = x_j(t+1) \cdot rtt_j(t+1)$. RateMP adjusts x_j and w_j per ACK. After receiving an ACK, the rate adjustment of a subflow j based on ECN is:

$$\delta_1 = \frac{\sum_j x_j}{x_j \cdot rtt_j}\left(\frac{\alpha_{total}}{\sum_j x_j} - \beta_j\right). \quad (10)$$

Similarly, based on delay detection, the adjustment of sending rate can be calculated as:

$$\delta_2 = \frac{\sum_j x_j}{x_j \cdot rtt_j}\left(\frac{\alpha_{total}}{\sum_j x_j} - \frac{\Delta rtt_j \cdot c}{K_{max}}\right). \quad (11)$$

When the sending rate is large (that is, $x_j \cdot rtt_j > 1$), RateMP only uses an ECN-detection to adjust the sending rate. That is $x_j \leftarrow x_j + \delta_1$. When the sending rate is small, RateMP uses cooperative congestion detection and rate adjustment to speed up the convergence to the network changes. Let $\delta = (\delta_1 + \delta_2)/2$ denote the average adjustment of sending rate based on ECN-delay combined congestion detection. If $\delta > 0$, RateMP utilizes δ_2 to speed up the increase function.

Otherwise, RateMP still utilizes δ_1 to adjust the sending rate and cwnd.

Algorithm 1: RateMP control loop

```
1  if Receive a new ACK then
2      if in slow start state then
3          w_j = w_j + 1;
4          pacing_rate = 0;
5          if β_j > α_total / Σ_j x_j or w_i > ss_i then
6              Exit slow_start state;
7      else
8          Update β_j;
9          Calculate Δrtt_j = new_rtt_j − base_rtt_j;
10         if x_j · rtt_j > 1 pkt then
11             x_j = x_j + (Σ_j x_j / (x_j · rtt_j)) (α_total / Σ_j x_j − β_j);
12         else
13             δ_1 = (Σ_j x_j / (x_j · rtt_j)) (α_total / Σ_j x_j − β_j);
14             δ_2 = (Σ_j x_j / (x_j · rtt_j)) (α_total / Σ_j x_j − Δrtt_j c / K_max);
15             δ = (δ_1 + δ_2)/2;
16             if δ ≤ 0 then
17                 x_j = x_j + δ_1;
18             else
19                 x_j = x_j + δ_2;
20         x_j = max{min_rate, x_j};
21         w_j = max{1, x_j · rtt_j};
22         if x_j · rtt_j > 1 pkt then
23             pacing_rate = 0;
24         else
25             pacing_rate = x_j;
```

D. Rate Control under Window Limitation

A hybrid window/rate control is used to reduce short-term burst traffic conflicts by eliminating the limit of the subflow minimum cwnd and using under-rate pacing. RateMP sets and controls the actual sending rate of each subflow at a low rate. Let $pacing\_rate$ denote the sending rate of a subflow, and RateMP set x_j to the $pacing\_rate$ of each subflow to control its sending rate. On the one hand, a small sending rate setting protects the incast scenarios from buffer float. On the other hand, for the scenarios where MPTCP should migrate data among subflows, a small minimum sending rate can leave other flows an available high bandwidth. However, this method will introduce the convergence problem at a low rate. When the sending rate is low, the incentive decreases due to the absence of feedback information (ECN marking) since it needs several ACKs to calculate the ECN marking probability. As a result, subflows cannot quickly fill the available bandwidth after the path congestion disappears. This problem is addressed by the cooperative congestion detection and convergence acceleration mechanism as mentioned above.

The hybrid window/rate control logic of RateMP is as follows. The sender employs a dynamic decision that adapts the window and rate control based on the current sending rate. When the transmission rate is high, the sender utilizes window control to ensure the prompt delivery of packets while minimizing control overhead. When the transmission rate is low, the sender switches to rate control.

In rate control, the sender utilizes a $pacing\_rate$ to control the actual sending rate of each subflow. Then, $pkt/pacing\_rate$ is set as the sending interval between two packets, where pkt is the size of a packet. At a low rate situation, the sending interval between two packets can be more than an RTT.

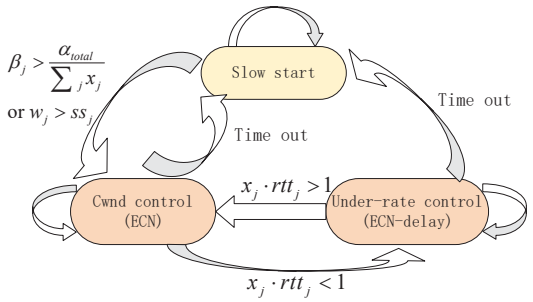


Fig. 2. Congestion control state transitions of RateMP.

Fig. 2 shows the state transition of RateMP. There are three states: slow start, cwnd control (ECN), and under-rate control (ECN-delay). At the beginning when a connection is established, RateMP enters a slow start state as traditional congestion control algorithms do to quickly detect the available bandwidth. In the slow start state, it uses window-based congestion control and the cwnd increases 1 when an ACK is received. If $\beta_j > \alpha_{total}/\sum_j x_j$ or $w_j > ss_j$, where ss_j is the slow start threshold, RateMP exits the slow start state and enters a cwnd control (ECN) state. Transitions between cwnd control (ECN) and under-rate control (ECN-delay) states are judged by $x_j \cdot rtt_j$. If $x_j \cdot rtt_j > 1$, RateMP enters the cwnd control (ECN) state and only uses ECN for rate adjustment. Otherwise, RateMP enters the under-rate control (ECN-delay) state and adds delay for accelerating the rate adjustment. Algorithm 1 shows the RateMP control loop. In line 20, RateMP sets $x_j = \max\{min\_rate, x_j\}$, which means setting a minimum sending rate as min_rate. The default min_rate is set to $0.01pkt/rtt_j$, referring to sending 0.01 packet per RTT.

IV. Analysis and Discussion

A. Equilibrium Analysis

Theorem 1. *RateMP control system is Lyapunav stable and asymptotically stable.*

Proof. We analyze the stability based on a widely used fluid model in [19]. Assume there is a lone-term flow transmit data with N subflows sharing a bottleneck link with bandwidth c.

Let $x_j(t)$ denote the window size of subflow j at time t, $q(t)$ denote the bottleneck queue length at time t. We have:

$$q(t+\delta t) - q(t) = \left(\sum_j x_j(t) - c \right) \delta t. \tag{12}$$

The adjustment logic of RateMP's control is:

$$x_j(t+\delta t) - x_j(t) = \sum_j x_j(t) \left(\frac{\alpha_{total}}{\sum_j x_j(t)} - \beta_j(t) \right) \delta t, \tag{13}$$

where $\beta_j(t)$ is the ECN mark proportion of received ACKs, and $\beta_j(t) = \alpha_{total} \cdot \eta q_j(t)$. Let $\eta' = \alpha_{total} \cdot \eta$, and the aggregate rate $x(t) = \sum_j x_j(t)$ can be expressed as:

$$\dot{x}(t) = x(t) \left(\frac{\alpha_{total}}{x(t)} - \eta' q_j(t) \right). \tag{14}$$

And the queue length can be expressed as:

$$\dot{q}(t) = x(t) - c. \tag{15}$$

When the sending rate and the queue length stabilize i.e., $\dot{x}(t) = 0$ and $\dot{q}(t) = 0$, there exists a unique equilibrium point $(x_e, q_e) = (c, \frac{\alpha_{total}}{\eta' c})$. By applying a change of variable from x_e to $x(t) = x_e + \delta x(t)$ and linearize Eq. (14) and Eq. (15) around (x_e, q_e), we have:

$$\delta \dot{x}(t) = -\eta' q_e \delta x(t) - \eta' x_e \delta q(t). \tag{16}$$

$$\delta \dot{q}(t) = \delta x(t). \tag{17}$$

Converting the above differential equations to a matrix form, we have:

$$\begin{bmatrix} \delta \dot{x}(t) \\ \delta \dot{q}(t) \end{bmatrix} = \begin{bmatrix} -\eta' q_e & -\eta' x_e \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \delta x(t) \\ \delta q(t) \end{bmatrix}. \tag{18}$$

The eigenvalues of the system are $\frac{-\eta' - \sqrt{\eta'^2 - 4\eta'}}{2}$ and $\frac{-\eta' + \sqrt{\eta'^2 - 4\eta'}}{2}$. Since $\eta' > 0$, both the eigenvalues have negative parts. This proves that the system is Lyapunav stable and asymptotically stable. □

B. Discussion

RateMP utilizes ECN and delay for in-network congestion notification and rate adjustment. The notification based on ECN requires a unified deployment of ECN on switches in the network, as well as ECN support on end nodes. Since ECN is a common protocol for switches and routers in networks, and is widely supported by modern routers and switches. It is easy to deploy the ECN mechanism on all switches in DCNs and set the ECN parameters uniformly on all switches. The notification based on delay only requires RTT detection on end nodes, which does not need other modifications to switches.

Moreover, due to the limitation of low rate, RateMP may result in significant disparities between subflows, leading to the occurrence of HoL blocking. To address this issue, RateMP can incorporate a packet scheduler as an extended patch, which receives data from applications and sends them to each subflow after segmenting and processing [20], [21].

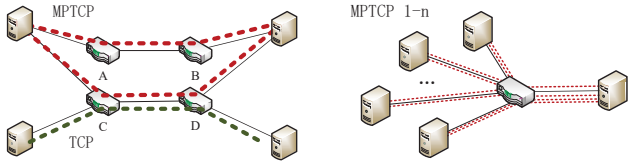
When a subflow enters the low rate control phase while other subflows are still in the cwnd control phase, indicating a substantial discrepancy between the subflows, the scheduler recognizes the need for additional measures. In such cases, the scheduler introduces redundant packet transmissions for the subflows in a low rate. This redundant data transmission helps alleviate the Head-of-Line (HoL) blocking problem by compensating for the excessive path variations. By introducing redundant packets, the scheduler ensures that the subflows experiencing lower rates do not block the whole transmission. And, it ensures that the subflows remain active, allowing them to immediately detect and utilize idle bandwidth whenever it becomes available on the paths. This approach enables RateMP to efficiently utilize in-network paths and maximize overall network throughput.

V. Performance Evaluation

In this section, we show the evaluation of RateMP in different network scenarios. We first implement RateMP in Linux kernel and evaluate it in small-scale networks. RateMP is implemented in MPTCP v0.95 [22] as a kernel module. The kernel module implements the MPTCP *ratemp_main*() and *ratemp_set_cwnd_and_pacing_rate*() functions to enforce its direct rate control and rate adjustment to subflows, respectively. We then show RateMP's performance through a large-scale simulation in NS-3 [23].

A. Kernel-based Evaluation

Fig. 3 shows the small-scale network testbed topology of multi-bottleneck and incast scenarios, where we evaluate and show the performance of data migration ability and delay. The ECN parameters used in RateMP are set to $K_{min} = 0$, $K_{max} = 150$ KB, $P_{max} = 1$. We set $\alpha_{total} = c/100$, where c is the link bandwidth. We perform the measurement and compare the results of RateMP with DCTCP, BALIA, and LIA. DCTCP is a single-path congestion control algorithm, we run it independently on each subflow.



(a) Multi-bottleneck (b) Incast

Fig. 3. Testbed setup.

Fig. 4 is the performance comparison under a topology shown in Fig. 3(a), which is a topology of multiple bottlenecks. The bandwidth of bottleneck links AB and CD is set to 0.5 Gbps, and the link delay is set to 100 μs. MPTCP flow starts at 0 s and continues transmitting data. TCP flow starts at 5 s and ends at 10 s. Fig. 4 shows the real-time throughput of different algorithms. RateMP can quickly achieve the optimal load-balancing effect, and maintains a stable sending rate. BALIA decides the state of subflows according to the lost

packet situation and then adjusts their cwnd, so the result is not as accurate as RateMP. Since the packet loss situation is relatively random, it sometimes misjudges and results in performance degradation. DCTCP and LIA have poor load-balancing ability. The throughput of TCP is low due to the unreasonable allocation of resources among subflows.

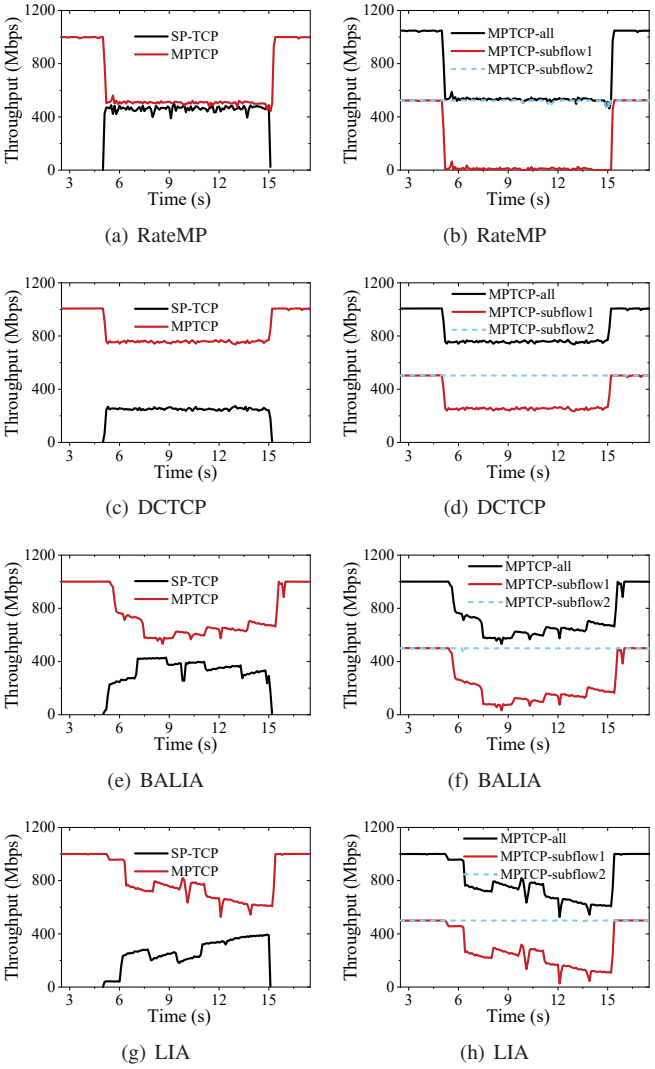


Fig. 4. Real time throughput of different congestion control algorithms.

TABLE I
RTT COMPARISON OF DIFFERENT ALGORITHMS

	Average RTT	RTT variance
RateMP	0.236 ms	0.0122 ms
DCTCP	0.615 ms	0.229 ms
BALIA	7.35 ms	3.14 ms
LIA	8.56 ms	2.75 ms

Table I shows the average RTT of each MPTCP flow of different congestion control algorithms. LIA and BALIA have the largest RTT with high fluctuation. The RTT of DCTCP decreases a lot, since it utilizes ECN for in-network congestion notification and keeps a low queue in the network. However, the basic RTT is 200 µs, and the RTT of DCTCP is still several times of the basic RTT, which is caused by the queue delay. RateMP keeps the lowest average RTT, which is very close to the basic RTT, and its RTT variance is also the lowest. Compared with other algorithms, RateMP reduces 50% RTT.

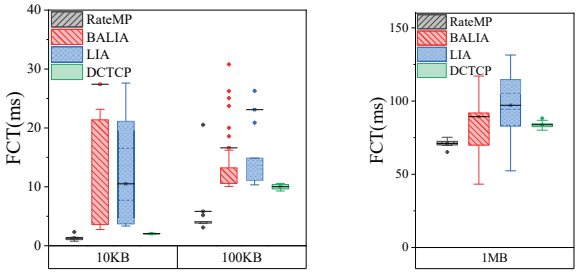


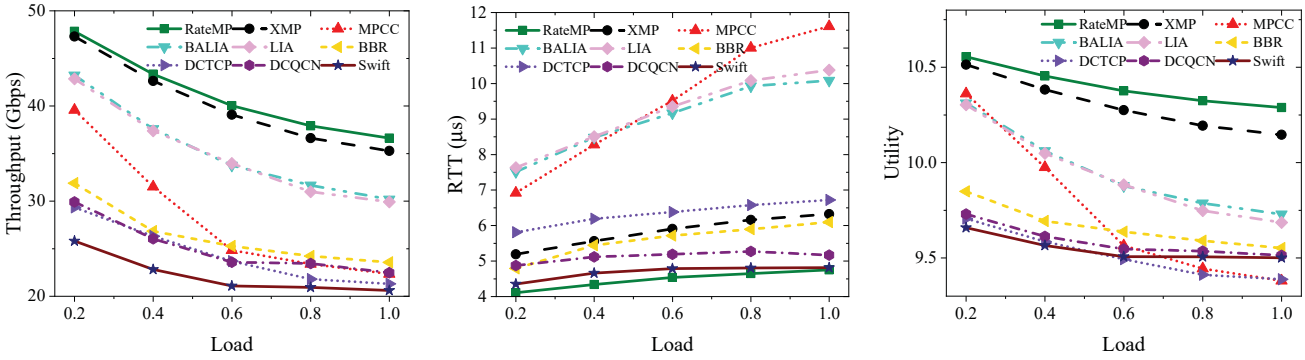
Fig. 5. FCT of different congestion control algorithms.

Fig. 3(b) is an incast scenario. The bandwidth of the bottleneck link is set to 1 Gbps, and the link delay is set to 100 µs. There are 8 long MPTCP flows, each has 2 subflows. And the right server downloads data with different sizes from the last server at the left. Fig. 5 shows the flow completion time (FCT) of downloading data with different sizes (10 KB, 100 KB, and 1 MB). Fig. 5(a) shows the FCT of small flows. RateMP provides the smallest FCT with low fluctuation. BALIA and LIA show large average FCTs with high fluctuation, since they make high queue delay and easily cause time-out. DCTCP also keeps little fluctuation in FCTs, but the average FCT is larger than RateMP. Especially, RateMP reduces FCTs more than 45% compared with other algorithms when transmitting data of 100 KB.

B. Simulation in Large-scale Networks

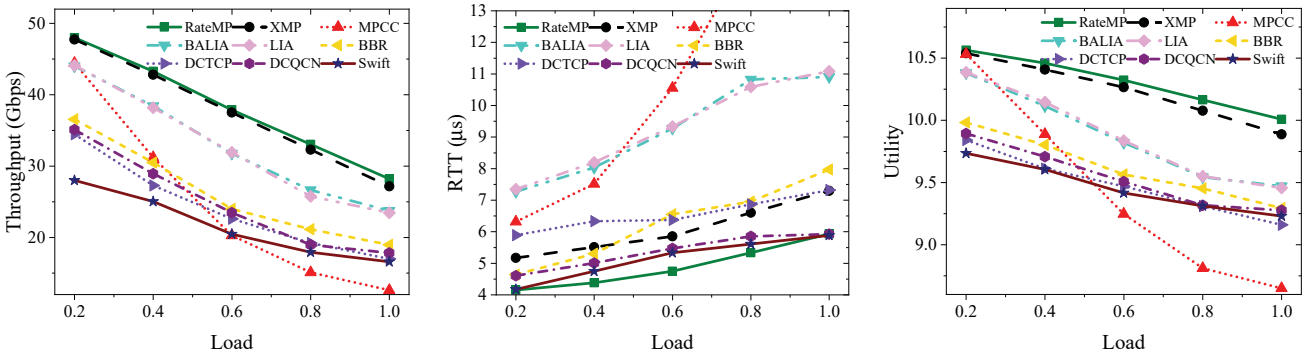
Setup. The DCN topology is a 3-tier Fat-Tree topology with parameter $K = 10$, including 125 switches and 500 servers. Link bandwidth between switches is 100 Gbps. The propagation delay between aggregation switches and core switches is 0.5 µs, and the remaining links are 0.1 µs. Switches use per-flow ECMP routing [24]. The buffer size of each switch is set to 4 MB. The RateMP parameters are set to $K_{min} = 0$, $K_{max} = 40$ KB, $P_{max} = 1$, $\alpha_{total} = c/100$, where $c = 100$ Gbps is the link bandwidth. LIA, BALIA, XMP, MPCC, DCTCP, DCQCN, BBR, and Swift are used for comparison. DCTCP, DCQCN, BBR, and Swift are single-path transmission protocols, which only use one path for each flow. We generate traffic based on cache follower traffic workload and DCTCP traffic workload.

Random traffic in the network. Fig. 6 shows the performance of random traffic. The traffic load increases from 0.2 to 1.0. The network traffic is set randomly, where each server sends to one other server chosen at random. Under the random traffic, MPTCP benefit for load balancing is obvious because it is able to use multiple paths to balance the load to every available link in the network, especially when the network load is light. When the network traffic



(a) Throughput of all flows (b) RTT of all flows (c) Utility of all flows

Fig. 6. Average throughput, RTT, and utility of different traffic load under a Fat-Tree topology.



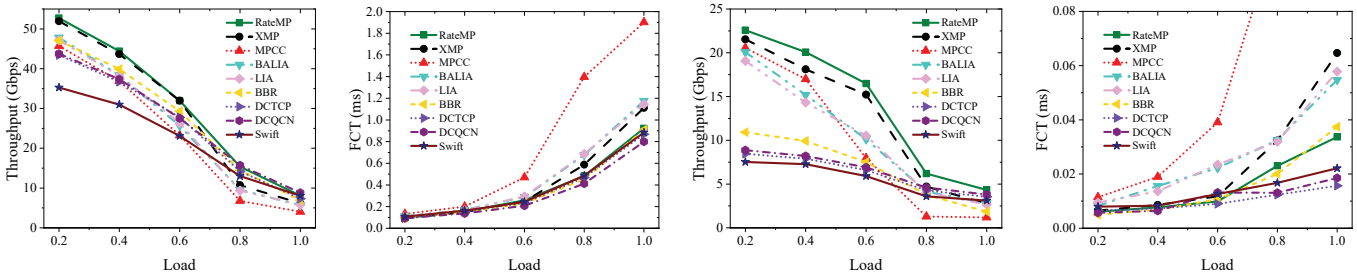
(a) Throughput of all flows (b) RTT of all flows (c) Utility of all flows

Fig. 7. Average throughput, RTT, and utility of many-to-many traffic under a Fat-Tree topology.

load is less than 0.4, MPTCP with RateMP, XMP, MPCC, BALIA, and LIA all achieve higher average throughput than single path protocols (TCP with DCTCP and BBR, RDMA with DCQCN and Swift). Among them, RateMP has the highest average throughput, XMP is the second highest which is close to RateMP, and the others have lower throughput. This is because both RateMP and XMP utilize ECN as a notification for in-network congestion feedback, which is more accurate for transmission adjustment. Among them, RateMP uses ECN to provide more accurate feedback on the degree of congestion in the network, while adding and delay for auxiliary judgment, so the throughput is the highest. And this advantage is more obvious when the network load is high. LIA and BALIA are loss-based algorithms continue to increase the cwnd until packet loss happens. Despite they working with RED set up at switches, they still cause relatively long delay and throughput loss. MPCC is a lightweight online learning algorithm that relies on the parameter configuration of the reward in practical. The parameter configuration used in the experiments is underlying the reference [25]. While MPCC is better suited to the heterogeneous Internet, it performs poorly in DCNs. With the increase of traffic load, the throughput of MPCC becomes even lower than that of single path protocols. Moreover, RateMP keeps the lowest RTT at around 4.5 μs, and its RTT grows slightly when the network traffic load increases.

Many-to-many traffic in the network. Then, we simulate a many-to-many traffic situation. In the Fat-Tree topology, only the hosts in the first port are chosen as receivers. At each time, we randomly choose a server as the sender and a server in the last port as the receiver to transmit data. RateMP, XMP, BALIA, and OLIA utilize 4 subflows in each MPTCP connection. DCTCP and DCQCN utilize a single path. Fig. 7 shows the performance of average throughput and RTT of different algorithms with changing traffic density. When there are small number of flows in the network, LIA and BALIA achieve higher throughput than DCTCP and DCQCN. With the increase of flows, the throughput of LIA and BALIA becomes less than that of DCTCP and DCQCN. Moreover, LIA and BALIA have the highest RTT. Since loss-based algorithms continue to increase the cwnd until packet loss happens, hence cause long delay. RateMP keeps the highest throughput with the lowest RTT. Compared with XMP, DCTCP and DCQCN, which also use ECN detection, the RTT of RateMP keeps low and increases slightly.

Incast traffic in the network. Fig. 8 shows the experimental results with an incast (many-to-one) traffic. In the Fat-Tree topology, each time it randomly chooses a server as sender, sends data to the last server. MPTCP utilizes 8 subflows in each connection. The background traffic in the whole network is set as 0.5, and the incast traffic load increases from 0.2 to



(a) Throughput of all flows (b) FCT of all flows (c) Throughput of small flows (d) FCT of small flows

Fig. 8. Performance of many-to-one traffic under a Fat-Tree topology.

1.0. RateMP always maintains the highest average throughput.

Fig. 8(a) and Fig. 8(b) are the average throughputs and FCTs of all flows in the network, respectively. RateMP keeps a higher average throughput than other algorithms. Loss-based congestion control algorithms (LIA, BALIA) increase the cwnd continuously, which results in the collective packet loss in network congestion and performs the worst. ECN-based congestion control algorithms (XMP, DCTCP, DCQCN) show better performance, however, still cause a degradation when the number of incast flows increases. RateMP keeps low queue occupancy through a low rate setting, reducing performance degradation caused by continuous packet loss. Fig. 8(c) and Fig. 8(d) are the average throughputs and FCTs of small flows in the network, respectively. With the increase of traffic load, the number of subflows (of MPTCP) converging at the last hop increases, causing long queues and intensive continuous packet loss. RateMP lowers the sending rate, therefore reducing queue occupancy and achieving smaller FCT than other MPTCP congestion control algorithms (LIA, BALIA, XMP, MPCC).

VI. Related Work

Multi-path congestion control algorithms: LIA [26] is the first algorithm that is designed to couple the congestion control strategies running on different subflows through the growth rate of correlation subflows. OLIA [27] and BALIA [28] further improve MPTCP's load balancing ability through different designs on window adjustment methods. Coupled BBR [29] and SBCC [30] are further proposed to enhance the performance of loss tolerance and shared bottleneck fairness. ECN-based algorithms, such as XMP [31] and AMP [32], provide better throughput and delay performance in DCNs. They use the same ECN-mark mechanism as DCTCP does to control buffer occupancy in the switch, and make a trade-off between latency and throughput through different designs on window adjustment. MMPTCP [33] uses a packet scattering technique to improve the delay performance of short flows while it acts as a regular MPTCP for long flows. In Internet, the change of network conditions is usually unpredictable, especially in heterogeneous and dynamic network environments. MPCC [25] designs a utility function apply it to convex optimization, and decides its sending rates through trial-and-error. MPCC shows excellent ability to dynamically make appropriate adjustments to changes in network conditions.

Single-path congestion control algorithms: 1) ECN-based: ECN-based proposals usually adopt ECN marking based on instant queue length, such as DCTCP [16], D2TCP [34], L2DCT [35], and DCQCN [36]. DCTCP is the most commonly used end-to-end transmission protocol in DCNs, which enables ECN to estimate congestion probability and adjust the cwnd in advance. D2TCP and L2DCT build upon DCTCP. D2TCP focuses on decreasing the likelihood of missed deadlines for TCP flows, and L2DCT aims to reduce FCT for short flows. DCQCN works like DCTCP depending on ECN to modulate sending rates but it is designed for RDMA. 2) Delay-based: TIMELY [37] and Swift [38] uses special network interface cards (NIC) to measure RTT in DCNs at the microsecond granularity, predict in-network congestion, and keep buffer occupancy low. 3) In-network telemetry (INT)-based: HPCC [39] and PowerTCP [40] utilize the INT to obtain detailed switch and link information [41]. These algorithms generally require switch or NIC configuration and computing power to break through traditional performance bottlenecks.

VII. Conclusion

In this paper, we analyzed the dilemma that MPTCP achieves high bandwidth utilization but hinders the performance of short-burst traffic in DCNs, and highlighted the limitation of existing coupled congestion control algorithms. To overcome this limitation, we proposed RateMP, which utilizes a hybrid control loop with fine-grained congestion degree detection to eliminate the limitations. We have implemented and evaluated RateMP, and shown that RateMP can greatly improve the performance of DCNs, reducing traffic delay by more than 40% in most cases with high throughput achievement.

Acknowledgment

This work is supported in part by the National Natural Science Foundation of China under Grant No. 62302472, Youth Innovation Promotion Association of the Chinese Academy of Sciences (CAS) under Grant No. Y202093, JSPS KAKENHI under Grant No. JP19H04105, and the Fundamental Research Funds for the Central Universities under grant No. WK2100000039.

REFERENCES

[1] S. Yan, X. Wang, X. Zheng, Y. Xia, D. Liu, and W. Deng, "ACC: Automatic ecn tuning for high-speed datacenter networks," in *Proceedings of the 2021 ACM Special Interest Group on Data Communication (SIGCOMM)*, 2021, pp. 384–397.

[2] J. Ros-Giralt, N. Amsel, S. Yellamraju *et al.*, "Designing data center networks using bottleneck structures," in *Proceedings of the 2021 ACM Special Interest Group on Data Communication (SIGCOMM)*, 2021, pp. 319–348.

[3] Y. Zhang, X. Nie, J. Jiang *et al.*, "BDS+: An inter-datacenter data replication system with dynamic bandwidth separation," *IEEE/ACM Transactions on Networking*, vol. 29, no. 2, pp. 918–934, 2021.

[4] L. Luo, K.-T. Foerster, S. Schmid, and H. Yu, "Splitcast: Optimizing multicast flows in reconfigurable datacenter networks," in *Proceedings of the 2020 Annual IEEE International Conference on Computer Communications (INFOCOM)*. IEEE, 2020, pp. 2559–2568.

[5] C. Guo, G. Lu, D. Li *et al.*, "BCube: a high performance, server-centric network architecture for modular data centers," in *Proceedings of the 2009 ACM Special Interest Group on Data Communication (SIGCOMM)*, 2009, pp. 63–74.

[6] A. Ford, C. Raiciu, M. J. Handley, O. Bonaventure, and C. Paasch, "TCP extensions for multipath operation with multiple addresses," 2020, RFC 8684, Accessed: Jan., 2024. [Online]. Available: https://rfc-editor.org/rfc/rfc8684.txt

[7] G. Chen, Y. Lu, Y. Meng *et al.*, "Fast and cautious: Leveraging multi-path diversity for transport loss recovery in data centers," in *Prceedings of the 2016 USENIX Annual Technical Conference (ATC)*, 2016.

[8] L. Li, K. Xu, T. Li *et al.*, "A measurement study on multi-path TCP with multiple cellular carriers on high speed rails," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication (SiGCOMM)*, 2018.

[9] J. Han, K. Xue, W. Wei, Y. Xing, J. Liu, and P. Hong, "Transparent multipath: Using double MPTCP proxies to enhance transport performance for traditional tcp," *IEEE Network*, vol. 35, no. 5, pp. 181–187, 2021.

[10] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving datacenter performance and robustness with multipath TCP," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 266–277, 2011.

[11] J. Hwang, A. Walid, and J. Yoo, "Fast coupled retransmission for multipath TCP in data center networks," *IEEE Systems Journal*, vol. 12, no. 1, pp. 1056–1059, 2016.

[12] S. Liu, H. Xu, L. Liu, W. Bai, K. Chen, and Z. Cai, "RepNet: Cutting latency with flow replication in data center networks," *IEEE Transactions on Services Computing*, 2018.

[13] G. Zeng, L. Chen, B. Yi, and K. Chen, "Cutting tail latency in commodity datacenters with cloudburst," in *Proceedings of the 2022 IEEE International Conference on Computer Communications (INFOCOM)*. IEEE, 2022, pp. 600–609.

[14] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[15] K. Ramakrishnan, S. Floyd, and D. Black, "The addition of explicit congestion notification (ECN) to IP," 2001, RFC 3168, Accessed: Jan., 2024. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8311

[16] M. Alizadeh, A. Greenberg, D. A. Maltz *et al.*, "Data center TCP (DCTCP)," in *Proceedings of the 2010 ACM Special Interest Group on Data Communication (SIGCOMM)*, 2010.

[17] Y. Zhu, M. Ghobadi, V. Misra, and J. Padhye, "ECN or delay: Lessons learnt from analysis of DCQCN and TIMELY," in *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2016, pp. 313–327.

[18] F. P. Kelly, A. K. Maulloo, and D. K. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, no. 3, pp. 237–252, 1998.

[19] V. Misra, W.-B. Gong, and D. Towsley, "Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED," in *Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, 2000, pp. 151–160.

[20] J. Han, K. Xue, J. Li *et al.*, "EdAR: An experience-driven multipath scheduler for seamless handoff in mobile networks," *IEEE Transactions on Wireless Communications*, vol. 22, no. 10, pp. 6839–6852, 2023.

[21] Y. Xing, K. Xue, Y. Zhang, J. Han, J. Li, and D. S. Wei, "An online learning assisted packet scheduler for MPTCP in mobile networks," *IEEE/ACM Transactions on Networking*, vol. 31, no. 5, pp. 2297–2312, 2023.

[22] "MultiPath TCP - linux kernel implementation," Accessed: Jan., 2024. [Online]. Available: http://multipath-tcp.org/

[23] "NS-3 simulator," Accessed: Jan., 2024. [Online]. Available: https://www.nsnam.org/

[24] C. Hopps, "Analysis of an equal-cost multi-path algorithm," Tech. Rep., 2000, RFC 2992, Accessed: Jan., 2024. [Online]. Available: https://www.rfc-editor.org/rfc/rfc2992

[25] T. Gilad, N. Rozen-Schiff, P. B. Godfrey, C. Raiciu, and M. Schapira, "MPCC: online learning multipath transport," in *Proceedings of the 16th International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2020, pp. 121–135.

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

[27] R. Khalili, N. Gast, M. Popovic, U. Upadhyay, and J.-Y. Le Boudec, "MPTCP is not pareto-optimal: performance issues and a possible solution," in *Proceedings of the 8th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT)*, 2012.

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

[29] J. Han, K. Xue, Y. Xing *et al.*, "Leveraging coupled BBR and adaptive packet scheduling to boost mptcp," *IEEE Transactions on Wireless Communications*, vol. 20, no. 11, pp. 7555–7567, 2021.

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

[31] Y. Cao, M. Xu, X. Fu, and E. Dong, "Explicit multipath congestion control for data center networks," in *Proceedings of the 2013 International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2013.

[32] M. Kheirkhah and M. Lee, "AMP: An adaptive multipath TCP for data center networks," in *Proceedings of the 2019 IFIP Networking Conference (IFIP Networking)*, 2019.

[33] M. Kheirkhah, I. Wakeman, and G. Parisis, "MMPTCP: A multipath transport protocol for data centers," in *Proceedings of the 35th Annual IEEE International Conference on Computer Communications (INFOCOM)*. IEEE, 2016, pp. 1–9.

[34] B. Vamanan, J. Hasan, and T. Vijaykumar, "Deadline-aware datacenter tcp (d2tcp)," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 115–126, 2012.

[35] A. Munir, I. A. Qazi, Z. A. Uzmi *et al.*, "Minimizing flow completion times in data centers," in *Proceedings of the 2013 Annual IEEE International Conference on Computer Communications (INFOCOM)*. IEEE, 2013, pp. 2157–2165.

[36] Y. Zhu, H. Eran, D. Firestone *et al.*, "Congestion control for large-scale RDMA deployments," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 523–536, 2015.

[37] R. Mittal, V. T. Lam, N. Dukkipati *et al.*, "TIMELY: RTT-based congestion control for the datacenter," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 537–550, 2015.

[38] G. Kumar, N. Dukkipati, K. J. Mpi-sws *et al.*, "Swift: Delay is simple and effective for congestion control in the datacenter," in *Proceedings of the 2020 ACM Special Interest Group on Data Communication (SIGCOMM)*, 2020, pp. 514–528.

[39] Y. Li, R. Miao, H. H. Liu *et al.*, "HPCC: High precision congestion control," in *Proceedings of the 2019 ACM Special Interest Group on Data Communication (SIGCOMM)*, 2019, pp. 44–58.

[40] V. Addanki, O. Michel, and S. Schmid, "PowerTCP: Pushing the performance limits of datacenter networks," in *Proceedings of the 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2022, pp. 51–70.

[41] W. Wang, M. Moshref, Y. Li *et al.*, "Poseidon: Efficient, robust, and practical datacenter CC via deployable INT," in *Proceedings of the 2023 USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2023.